MASTER EN SCIENCES INFORMATIQUES

Prédiction des séries temporelles: applications aux trajectoires balistiques

Dallons, Gauthier; Triffin, François

Award date:
2004

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année Académique 2003-2004

Prédiction de séries temporelles : Application à la prédiction de trajectoires balistiques

Gautier Dallons, François Trifin

Mémoire présenté en vue de l'obtention du grade de
Maître en Informatique .

Résumé

La prédiction de séries temporelles est étudiée dans beaucoup de domaines. Ainsi, elle fait l'objet de recherches en économie, en mathématique ou encore en robotique. Dans cette dernière branche, la prédiction de séries temporelles a pour mission de pouvoir faire réaliser des tâches plus complexes par des robots en y incluant le temps. C'est à cela que le groupe "TROP" a décidé récemment de s'investir.

Pour aborder le problème, ce mémoire étudiera la prédiction d'une classe de séries temporelles particulières : les séries temporelles balistiques. Cette classe est intéressante car elle pourrait permettre à un robot d'attraper, par exemple, une balle. Ce mémoire évaluera différentes méthodes de prédiction des séries temporelles balistiques : des approches neuro-mimétiques, des approches de type ingénierie et des approches de type régression linéaire ou polynômiale.

Mots-clés : séries temporelles, prédiction balistique, robotique, traitement du signal adaptatif, réseaux de neurones, ingénierie, régression linéaire, régression polynômiale.

Abstract

Time series prediction is studied in many area's, such as economics, mathematics and robotics. In robotics, the goal of the time series prediction is to have robots accomplish more complicated tasks including time. The group "TROP" has decided to investigate in this way.

To tackle this problem, this master's thesis will focus on a particular set of time series : the ballistic time series. This problem set is interesting because it might enable a robot to catch a ball, for example. This master's thesis will evaluate different prediction methods : neural networks, engineering, and approaches like linear regression and polynomial regression.

Keywords : time series, ballistic prediction, robotics, adaptive signal processing, neural networks, engineering, linear regression, polynomial regression.

Nous tenons à remercier particulièrement les professeurs J.-L.Buessler et J.-P. Urban pour l'accueil chaleureux dans leur laboratoire pendant les 5 mois de notre stage. Nous voulons leur faire part de notre gratitude pour le temps qu'ils nous ont consacré ainsi que pour leurs très nombreux conseils avisés.

Nous remercions notre promoteur, J.-P. Leclerq, pour ses encouragements pendant notre stage et pour la lecture critique et constructive de notre mémoire.

Nous remercions chaleureusement nos familles respectives et Aurélie Van Oppens pour leur soutien et les relectures de ce mémoire.

Enfin, nous remercions, Damien, Djaffar, Gilles, Laurentiu, Patrice et tous les autres membres de l'équipe du laboratoire "TROP" pour tous les nombreux moments inoubliables passés en leur compagnie.

Table des matières

Glossaire xv

Introduction 1

1 Contexte de la problématique 5

1.1 Introduction 5
1.2 Description de la plate-forme robotique 5
1.3 Problématique en rapport avec le groupe "TROP" 7
1.4 Prédiction de séries temporelles 8
1.5 Prédiction de séries temporelles issues d'un système dynamique théorique particulier 10
1.6 Prédiction de trajectoires balistiques 12
1.7 Critique 13
1.8 Problématique en rapport avec la littérature 15
1.9 Conclusion 15

2 Réseaux de neurones 17

2.1 Introduction 17
2.2 Contexte historique 17
2.3 Modélisation d'un neurone biologique 20
2.3.1 Introduction 20
2.3.2 Formalisation 21
2.4 Présentation de l'élément adaline 24
2.5 Apprentissage de l'adaline 24
2.5.1 Introduction 24
2.5.2 Erreur et performance 25
2.5.3 Solution directe 27
2.5.4 Descente de gradient instantané 27
2.5.5 L'algorithme Least Mean Square (LMS) 28
2.5.6 L'algorithme alpha-LMS (α-LMS ou NLMS) 31
2.5.7 L'algorithme Recursive Least Square (RLS) 32
2.6 Architectures des réseaux de neurones 34
2.7 Caractéristiques des réseaux de neurones 35
2.8 Conclusion 38

3 Prédiction balistique dans l'espace cartésien **39**
3.1 Introduction 39
3.2 Modèle de prédiction balistique simple 39
3.2.1 Hypothèses 39
3.2.2 Modèle d'une trajectoire sans force de frottement 40
3.2.3 Principe théorique de la discrétisation 40
3.2.4 Modèle neuro-mimétique d'apprentissage d'une trajectoire sans force de frottement 42
3.2.5 Validation du modèle neuro-mimétique sur base de données simulées avec $Matlab_{©}$ 46
3.2.6 Validation théorique du modèle neuro-mimétique 49
3.3 Modèle de prédiction balistique avec frottement 53
3.3.1 Hypothèses 53
3.3.2 Modèle d'une trajectoire avec force de frottement 53
3.3.3 Modèle neuro-mimétique d'apprentissage d'une trajectoire avec force de frottement 54
3.3.4 Résultats des simulations sur base de données simulées avec $Matlab_{©}$ 55
3.3.5 Analyse de sensibilité du modèle à la force de frottement de l'air 60
3.4 Conclusion 62

4 Vision stéréoscopique **63**
4.1 Introduction 63
4.2 Description du problème 63
4.3 Modélisation d'une caméra perspective 64
4.4 Description du cadre d'expérimentation 67
4.4.1 Cadre général 67
4.4.2 Calcul de la profondeur 68
4.4.3 Informations techniques sur le dispositif 70
4.5 Conclusion 71

5 Traitement d'images **73**
5.1 Introduction 73
5.2 Détection d'objet 73
5.2.1 Principe général 73
5.2.2 Algorithme de détection 73
5.3 Post-synchronisation des caméras 78
5.4 Réalisation 79
5.5 Conclusion 82

6 Modèle linéaire de prédiction balistique - Caméras parallèles **83**

6.1 Introduction 83
6.2 Hypothèses 83
6.3 Modèle neuro-mimétique d'apprentissage balistique 85
6.4 Calcul de $\vec{\Upsilon}^T(t-h)$ 89
6.5 Validation du modèle avec données simulées avec le logiciel $Matlab_{©}$. 92
6.6 Résultats sur des données réelles 94
6.7 Conclusion 94

7 Modèle de prédiction balistique - Caméras non-parallèles **97**

7.1 Introduction 97
7.2 Hypothèses 97
7.3 Méthode 97
7.4 Adaline propre à une trajectoire 100
7.5 Adaline utile à l'adaptation des coefficients de la fonction théorique de profondeur 102
7.6 Prédiction à l'aide d'un polynôme 104
7.7 Dispositions pratiques des simulations 104
7.8 Résultats des simulations 105
7.9 Analyse de sensibilité au bruit de la fonction de profondeur et de l'adaline propre à la trajectoire 106
7.10 Conclusion 107

8 Analyse des sources des problèmes **109**

8.1 Introduction 109
8.2 Bruit de pixélisation et de traitement d'images 109
8.3 Désynchronisation des deux images 111
8.4 Conséquences 115
8.5 Conclusion 115

9 Modèle polynômial de prédiction dans l'image **117**

9.1 Introduction 117
9.2 Principes du modèle 117
9.3 Régression polynômiale 118
9.4 Méthode 120
9.5 Résultats des simulations 122
9.6 Mise en oeuvre 123
9.7 Conclusion 124

10 Modèle multi-polynômial de prédiction dans l'image **125**
10.1 Introduction 125
10.2 Hypothèses 125
10.3 Méthode 126
10.4 Apprentissage des coefficients de la trajectoire dans l'image 126
10.5 Régression linéaire multiple 130
10.6 Calcul de prédictions 131
10.7 Résultats de simulations sur des données réelles 132
10.8 Conclusion 134

Conclusion **135**

Annexes **137**

A Modèle d'un objet uniformément accéléré **137**
A.1 Introduction 137
A.2 Développement 137

B Passage en temps discret **141**
B.1 Introduction 141
B.2 Développement 141

C Développement du calcul de la profondeur **143**
C.1 Introduction 143
C.2 Développement 144

Bibliographie **148**

Index **153**

Table des figures

1.1 Schéma de la plate-forme robotique. 6
1.2 Méthode générale de prédiction d'une série temporelle. 8
1.3 Système dynamique. 11
1.4 Méthode des trois phases. 12
1.5 Méthode des deux phases. 12

2.1 Représentation du système nerveux. 20
2.2 Neurone biologique. 21
2.3 Neurone formel. 22
2.4 Fonctions d'activation. 23
2.5 Surface de performance MSE 26
2.6 Interprétation géométrique du LMS 29
2.7 Surfaces de performance instantanée 30
2.8 Réseau simple-couche acyclique. 34
2.9 Réseau multi-couches acyclique. 35
2.10 Réseau récurrent 36

3.1 Modèle uniformément accéléré d'un projectile. 40
3.2 Système dynamique. 41
3.3 Architecture des réseaux d'adalines. 45
3.4 Prédiction à long terme. 46
3.5 Réponses des adalines sur la trajectoire de validation. 47
3.6 Erreur des adalines. 48
3.7 Evolution des coefficients des matrices des adalines 49
3.8 Modèle d'un projectile avec force de frottement. 54
3.9 Réponses des adalines sur la trajectoire de validation. 56
3.10 Erreur des adalines. 57
3.11 Erreur des adalines à long terme. 58
3.12 Evolution des coefficients des matrices des adalines. 59
3.13 Evolution des matrices des adalines (avec la force de frottement multipliée). 61

4.1 Schéma du modèle sténopé d'une caméra 65
4.2 Projection de l'intersection à l'infini des rails parallèles 65
4.3 Représentation schématique de la scène 67
4.4 Représentation géométrique de la scène dans le plan optique 68

4.5 Schéma du plan optique avec les cônes de vision des caméras 69
4.6 Comparaisons de la déformation de deux lentilles 70

5.1 Organigramme avec images de l'algorithme de détection d'objet 74
5.2 Organigramme de l'algorithme de post-synchronisation 80
5.3 Vision d'une caméra d'une suite d'échantillons avec le stroboscope . . 81
5.4 Vision d'une caméra d'une suite d'échantillons sans le stroboscope . . 81

6.1 Schéma de la scène vue par le haut 84
6.2 Schéma de la scène vue par le côté 84
6.3 Architecture des réseaux d'adalines 88
6.4 Résultats du modèle. 92
6.5 Convergence du modèle. 93
6.6 Erreur de prédiction du modèle. 94
6.7 Convergence du modèle sur des données réelles. 95
6.8 Résultats du modèle sur une trajectoire réelle. 95
6.9 Erreur du modèle sur une trajectoire réelle. 96

7.1 Schéma de la scène vue du haut. 98
7.2 Schéma de la scène vue du côté. 98
7.3 Modèle de prédiction 99
7.4 Adaline propre à une trajectoire. 102
7.5 Adaline d'apprentissage des coefficients de la fonction de profondeur. . 103
7.6 Résultats du modèle. 105
7.7 Erreur du modèle. 106
7.8 Analyse de sensibilité. 107

8.1 Pixélisation d'une balle proche de la caméra. 110
8.2 Pixélisation d'une balle éloignée de la caméra. 110
8.3 Erreur de mesure du centre de gravité. 110
8.4 Intégration d'une image par une webcam 111
8.5 Traînée de la balle dans l'image 112
8.6 Situation de deux webcams synchronisées. 113
8.7 Situation de deux webcams désynchronisées. 114

9.1 Variation des coordonnées d'une trajectoire réelle. 118
9.2 Modèle polynômial 121
9.3 Prédictions dans l'image de la caméra 1. 122
9.4 Prédictions dans l'image de la caméra 2. 123
9.5 Erreur de prédiction. 124

10.1 Vue de la scène par le haut. 125
10.2 Vue de la scène par le côté. 126
10.3 Méthode de prédiction. 127
10.4 Prédictions pour la caméra 1. 132
10.5 Prédictions pour la caméra 2. 133
10.6 Erreur de prédiction. 133

A.1 Modèle uniformément accéléré d'un projectile. 139

C.1 Représentation schématique de la scène 144
C.2 Représentation géométrique de la scène dans le plan optique 145
C.3 Illustration du raisonnement. 145

Glossaire

A

$\mathcal{A}$	Ensemble d'apprentissage du réseau.
α_1	Angle entre la base et l'axe optique de la caméra n°1.
α_2	Angle entre la base et l'axe optique de la caméra n°2.

C

C_1	Centre optique de la première caméra.
C_2	Centre optique de la deuxième caméra.

D

Δ_z	Différence de profondeur entre deux observations.
d_k	Sortie désirée du réseau.

E

ϵ_k	Erreur instantanée du réseau.

F

f	Distance focale des caméras, c'est-à-dire la distance en mètres entre le centre optique et le plan image.
$\mathcal{F}$	Fonction d'apprentissage.

G

$\vec{g}$	Vecteur de la force gravifique.

H

h	Taux d'échantillonnage.

J

J	Fonction de performance MSE.

L

l	Distance en mètres entre les deux centres optiques C_1 et C_2.

P

$\mathcal{P}$	Fonction de projection perspective.
ϕ	Matrice relative au principe de discrétisation.
$p_{Im}(t)$	Position du mobile dans l'image au temps t.
Ψ_x	Largeur d'un pixel en mètres.
Ψ_y	Hauteur d'un pixel en mètres.
φ	Fonction d'activation d'un neurone.
$\vec{p}_0$	Position de départ du mobile.
$\vec{p}(t)$	Position du mobile à l'instant t.

R

R	Matrice d'autocorrélation.
$\varrho(t)$	Disparité à l'instant t, c'est-à-dire la différence des observations des deux caméras à l'instant t.

U

u_{max}	Largeur en pixels de la résolution de l'image.
$u(t)$	Entrée du système au temps t.

V

$\vec{v}_0$	Vitesse de départ du mobile.
$\vec{v}(t)$	Vitesse du mobile à l'instant t.
$\vec{x}(t)$	Vecteur d'état du système au temps t.
$v_{Im}(t)$	Vitesse du mobile dans l'image au temps t.
v_{max}	Hauteur en pixels de la résolution de l'image.

W

W_i	Matrice des coefficients initiaux du réseau.
W_i^{new}	Matrice des coefficients du réseau après apprentissage.

Y

$y(t)$	Sortie du système au temps t.

Introduction

Pour réaliser ce mémoire, nous avons été accueillis au sein du groupe "TROP" du laboratoire "MIPS" (Modélisation Intelligence Processus Systèmes) de l'université de Haute Alsace à Mulhouse. Ce groupe se compose d'une dizaine de personnes. Parmi elles, on trouve des professeurs, des maîtres de conférence, des étudiants thésards et des étudiants préparant un DEA. A l'aide d'approches neuro-mimétiques, le groupe "TROP" est devenu expert dans l'asservissement visuel (le contrôle à partir d'informations visuelles) de processus robotiques. Actuellement, ce groupe de recherche tente d'étendre ses compétences de traitement d'images et son champ exploratoire, notamment en abordant le domaine des séries temporelles.

Notre sujet "Prédiction de trajectoires balistiques" est un sujet exploratoire pour le groupe "TROP". Il s'inscrit dans la prolongation de ses recherches actuelles. En effet, pour le moment, le bras de sa plate-forme robotique peut se déplacer à un endroit donné en apprenant l'espace qui lui est accessible. Cet espace est appelé l'espace de travail. Cependant, l'information liée au temps n'est pas encore exploitée. Dès lors, l'algorithme actuel n'est pas bien adapté pour, par exemple, intercepter une cible en mouvement. Nous décrirons plus précisément dans le premier chapitre, la position de notre travail par rapport à la plate-forme robotique du laboratoire et aussi par rapport à la littérature existante dans ce domaine.

A travers ce mémoire, nous tenterons de voir quels sont les moyens qui permettraient de prédire des trajectoires balistiques dans l'image de caméras stéréoscopiques. Ce problème peut être considéré comme un problème de prédiction de séries temporelles, comme un problème d'identification de système dynamique ou encore comme un problème d'ingénierie. C'est pourquoi, dans le premier chapitre, nous allons parcourir également la littérature existante dans ces trois domaines.

Nous présenterons, ensuite, les éléments théoriques qui nous serviront de support tout au long de ce mémoire. Ainsi, nous introduirons brièvement les réseaux neuronaux, les notions de couches et d'apprentissage. Enfin, nous étudierons en détail l'élément adaline ("ADAptive LINear Element") ainsi que les algorithmes liés à son apprentissage.

Dans le chapitre 3, nous tenterons d'appliquer une approche de type modélisation de système dynamique. Nous partirons du problème en le simplifiant et en le considérant dans un espace cartésien. Nous essaierons de nous familiariser avec l'apprentissage de modèles adalines pour réaliser la prédiction de la trajectoire dans cet espace. Ce

chapitre nous permettra de tirer un certain nombre d'enseignements sur la technique d'apprentissage de l'adaline mais aussi sur les problèmes de prédiction à long terme. Ces différents éléments seront utiles dans le reste du rapport.

Dans l'idée de se rapprocher du cadre d'expérimentation de la plate-forme, nous introduirons la vision perspective de caméras. Le problème sera, dès lors, abordé dans les images de deux caméras stéréoscopiques. Pour cela, nous aurons besoin d'étudier le modèle sténopé décrivant une caméra "trou d'épingle". Ce sujet sera traité dans le chapitre 4.

Puis, nous serons amenés à acquérir des données réelles pour pouvoir confronter à la réalité les modèles élaborés dans les chapitres suivants. Nous devrons donc considérer le problème de traitement d'images et celui de l'acquisition de trajectoires balistiques. Le chapitre 5 s'acquittera de cette tâche.

Nous tenterons, dans le chapitre 6, de prédire les trajectoires en considérant les caméras parallèles. Nous utiliserons des approches de type modélisation de systèmes dynamiques avec un simple réseau constitué de plusieurs adalines. Nous élaborerons, à cette fin, un modèle de prédiction que nous confronterons à la réalité. A l'aide des enseignements dégagés de cette confrontation, nous pourrons envisager une approche différente : une approche mixte de type ingénierie qui essaie d'apprendre les coefficients de la fonction de perspective.

Cette approche mixte sera réalisée en faisant intervenir, d'une part l'adaline, d'autre part un polynôme. Elle tentera de concilier la vision "régression polynômiale" avec la vision d'apprentissage de la configuration de la scène. Cette approche, développée dans le chapitre 7, sera confrontée à la réalité et nous serons amenés à l'invalider dans un contexte d'application réel.

Le chapitre suivant représentera un chapitre charnière. Il essaiera d'élucider les causes qui provoquent les faibles performances de ces modèles dans la réalité. Pour cela, nous étudierons les problèmes liés à l'acquisition des données par le traitement d'images, les limitations engendrées par le matériel technique et la conséquence de ces problèmes sur la prédiction dans l'image. Suite à cette analyse, nous créerons un nouveau modèle qui devra tirer enseignement des éléments révélés dans ce chapitre.

Ainsi, nous évaluerons la technique de régression polynômiale dans l'image : celle-ci considère des caméras non parallèles et complètement indépendantes entre elles. Nous verrons, dans le chapitre 9, si cette technique est satisfaisante dans un contexte d'application pratique. Cette approche sera aussi confrontée à la réalité. Elle montrera des limites d'application dans le contexte propre à la plate-forme du laboratoire.

C'est pourquoi, nous développerons, dans le chapitre 10, une méthode basée sur la division de deux polynômes. Cette méthode sera élaborée dans un but d'appli-

cation pratique nécessitant des contraintes fortes de rapidité de prédiction en vue, éventuellement, de pouvoir exploiter cette approche sur la plate-forme robotique.

Nous terminerons ce mémoire par une conclusion sur nos recherches. Celle-ci évaluera notre travail en fonction des objectifs que nous fixerons au chapitre 1. Elle proposera aussi des perspectives de recherche, sur base de ce rapport, pour le laboratoire "TROP".

Chapitre 1

Contexte de la problématique

1.1 Introduction

Ce chapitre a pour but de situer notre sujet de mémoire dans son contexte. Premièrement, il rentre dans les thèmes de recherches abordés par le groupe "TROP". Dès lors, la problématique est directement dépendante de la plate-forme robotique. Ensuite, comme le thème de notre mémoire a été abordé dans la littérature selon des formes plus ou moins proches, nous tenterons de situer notre sujet par rapport à la plate-forme robotique mais aussi par rapport à la littérature.

1.2 Description de la plate-forme robotique

La plate-forme robotique du laboratoire "TROP" [6, 47] se compose de plusieurs éléments : un système de vision, un bras robotique et un ensemble d'ordinateurs. Ces différents éléments remplissent des rôles particuliers et sont décrits dans les paragraphes suivants. La plate-forme est représentée schématiquement à la figure 1.1.

Le système de vision est constitué de deux caméras stéréoscopiques. Ces deux caméras ont pour but de fournir des informations sur les éléments intéressants de la scène. Le système est composé de deux caméras afin d'obtenir une certaine information de profondeur par rapport à ce système de vision. Cette information n'aurait pas été disponible avec un système de vision composé d'une seule caméra. Ainsi, il est possible de connaître la position de l'effecteur (l'extrémité du bras robotique) dans un espace qui n'est pas forcément l'espace de la scène reconstruit. Le système de vision est mobile et dispose de trois degrés de liberté.

Le bras robotique se meut selon six degrés de liberté. L'extrémité du bras est constituée de l'effecteur. Cet effecteur peut atteindre un ensemble de points de l'espace de la scène. Cet ensemble est assez restreint. En effet, de part les caractéristiques du bras, l'espace accessible est limité par les positions possibles du bras. Cet espace est appelé espace de travail. En dehors de cet espace, le bras est incapable d'atteindre un point particulier.

Pour permettre à ce système de fonctionner, la plate-forme est composée d'un ensemble d'ordinateurs. Un système distribué est responsable des différents traitements et des différentes commandes liés à la plate-forme. Ainsi, une partie de ce système

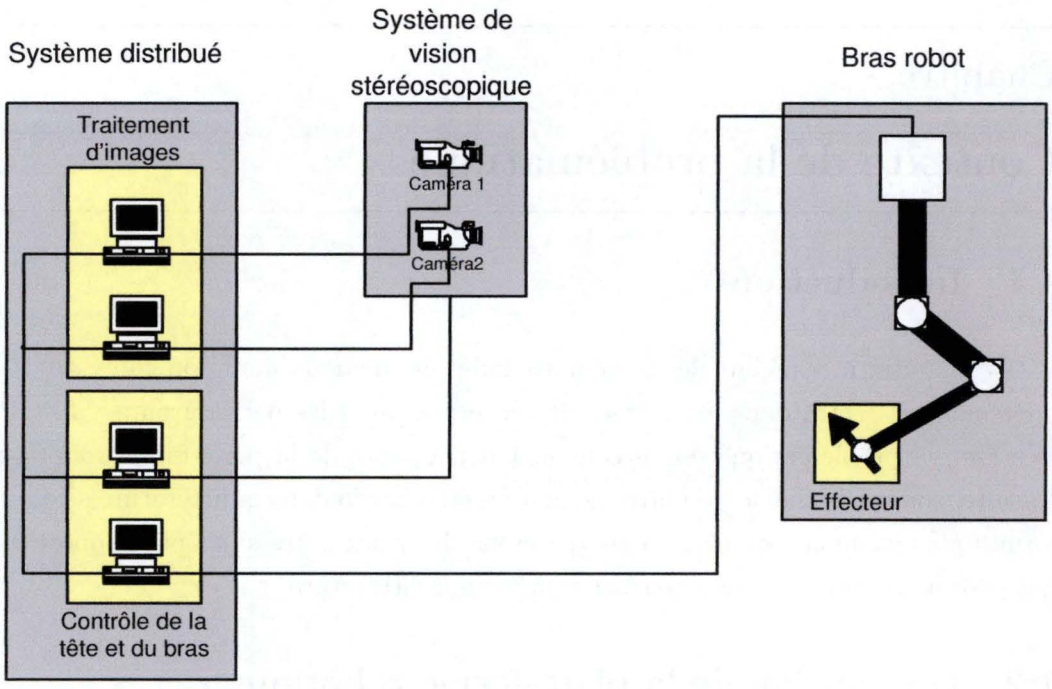


FIG. 1.1 – *Représentation schématique des composants de la plate-forme robotique.*

distribué sera responsable du traitement d'images, une autre de l'algorithme qui permettra au bras de se déplacer,... Ce système est soumis à des contraintes de temps réel fortes. En effet, il faut pouvoir traiter les informations d'une image avant que l'image suivante ne soit disponible et effectuer le déplacement correspondant. Cette problématique dépend, notamment, du taux d'échantillonnage du système de vision.

La vision constitue un élément clef de la plate-forme. En effet, les systèmes robotiques usuels utilisent des capteurs simples fournissant des informations élémentaires. La plate-forme du laboratoire "TROP" exploite, quant à elle, un système de vision stéréoscopique. Ce type de vision fournit une information bien plus riche. Cependant, une contrainte importante reste le temps de traitement de cette information. Il n'est pas encore envisageable de pouvoir exploiter, en temps réel, toute l'information disponible dans les images. En pratique, on exploitera des données ciblées. Par exemple, on extraira la position de l'effecteur dans les images et l'on définira une région d'intérêt (ROI) autour de cette dernière position pour le traitement de l'image suivante. Cet effecteur sera, par exemple, muni d'une lampe afin de simplifier le traitement d'images.

Le contrôle du bras robot par la vision s'appelle l'asservissement visuel. Il consiste à créer une boucle de rétroaction entre la vision et le déplacement du bras robotique. Cette rétroaction nécessitera de pouvoir identifier les paramètres articulaires du bras pour une position particulière d'une cible dans l'image. Ce problème est traité par

une fonction qui peut, éventuellement, être connue ou que l'on peut apprendre à l'aide d'approches telles que les approches neuro-mimétiques.

Ainsi, une approche neuro-mimétique simple que nous avons étudiée en détail, mais que nous ne développerons pas dans ce document, est constituée d'une carte auto-organisatrice de Kohonen (carte SOM) pour discrétiser l'espace et ainsi utiliser localement des modèles adalines pour permettre une adaptation locale [34].

Le laboratoire "TROP" tente d'évaluer et de tester différentes approches neuro-mimétiques dans l'asservissement visuel. Ces approches doivent être utilisées quand cela est nécessaire. L'objectif est de pouvoir se passer, autant que possible, d'informations sur la configuration de la scène et de ne pas avoir recours à une calibration (avec grille de calibration, ...).

1.3 Problématique en rapport avec le groupe "TROP"

Le sujet de notre mémoire consiste à étudier la prédiction de séries temporelles en rapport avec le contexte du laboratoire "TROP". Dans le cadre de la plate-forme robotique, le système de vision nous fournit des informations temporelles. Afin de réaliser cette étude, nous allons aborder un problème pratique de séries temporelles perçu dans les images : la prédiction de trajectoires balistiques. La problématique exploratoire et le contexte du laboratoire procureront à ce problème un aspect original pour plusieurs raisons.

Premièrement, la prédiction doit se réaliser dans l'espace des images des deux caméras du système stéréoscopique, car le bras robotique se déplace directement selon des coordonnées exprimées dans ces "espaces images". Il est donc intéressant de réaliser la prédiction de la trajectoire balistique dans cet espace afin d'éventuellement pouvoir intercepter un projectile.

Deuxièmement, cette prédiction devra être calculée en n'utilisant aucune ou peu d'informations sur la configuration de la scène (distance entre les caméras, ...). Ainsi, cette problématique s'inscrit tout à fait dans le contexte du laboratoire qui vise à apprendre un asservissement visuel indépendamment des caractéristiques de la scène.

Enfin, nous tenterons d'utiliser, quand cela est possible, des approches neuro-mimétiques. En effet, il est intéressant de pouvoir évaluer ces approches dans le traitement de la problématique de prédiction de trajectoires balistiques. Cependant, si cela est nécessaire, nous nous tournerons vers d'autres approches.

Maintenant, il est utile de parcourir la littérature et de situer notre sujet par rapport à certains thèmes de recherches d'autres laboratoires. Nous allons examiner trois grands axes : la prédiction de séries temporelles, la prédiction de séries temporelles

issues d'un système dynamique particulier et la prédiction de trajectoires balistiques. Une fois encore, nous situerons notre sujet par rapport à ces trois axes et nous verrons les éléments particuliers que nous voudrions apporter par rapport à l'existant.

1.4 Prédiction de séries temporelles

Pour prédire une série temporelle $y(1)$,$y(2)$,...,$y(n)$,... à l'instant suivant, on utilise une fenêtre temporelle d'entrée constituée des observations de cette série aux instants précédents. Cette fenêtre est caractérisée par sa taille, t, et l'intervalle de temps, (le délai) d, entre les observations la constituant. A partir de cette fenêtre d'entrée, une technique particulière va prédire l'élément suivant de la série (figure 1.2). A chaque nouvelle observation de la série, on peut constituer une nouvelle fenêtre d'entrée. Cela permet de calculer la prédiction suivante. Le pouvoir explicatif de ces techniques est peu important. En effet, un modèle est retenu selon sa capacité à prédire correctement et non sur l'interprétation que l'on peut en tirer pour valider celui-ci.

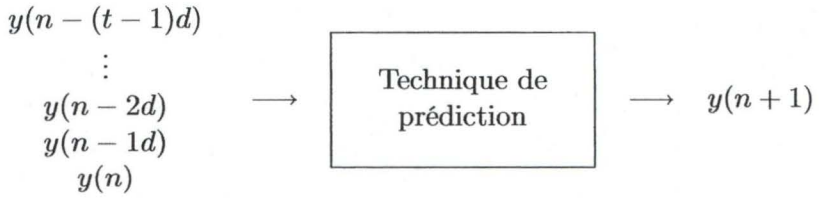


FIG. 1.2 – *Cette figure représente la méthode générale de prédiction d'une série temporelle d'une seule dimension en utilisant une fenêtre d'entrée constituée d'éléments de la série déjà observés. Cette méthode est facilement extensible aux vecteurs.*

Les techniques utilisées pour réaliser ces prédictions sont nombreuses et variées. Cependant, nous pouvons les classer selon trois catégories : les techniques à modèle unique, les techniques à modèles locaux et les techniques à modèles multiples. Le tableau 1.1 présente une synthèse de ces techniques.

Les techniques à modèle unique regroupent les réseaux neuronaux de type "rétro-propagation" [37, 24], les réseaux neuronaux "radial basis-function" (RBF) [7, 43], les réseaux "Finite Impulse Response" (FIR) [8] et les techniques de régressions habituelles connues sous les noms de ARMA, ARIMA et ARX [18]. Dans la vision à modèle unique, on met en oeuvre un seul modèle pour réaliser toutes les prédictions. Cela rend, généralement, le modèle ainsi conçu plus complexe et difficilement interprétable.

Les techniques à modèles locaux ont pour but de réaliser des approximations locales d'une fonction complexe. Elles sont constituées par des procédés faisant intervenir, en première couche, une carte auto-organisatrice de Kohonen (carte SOM) pour discréti-

ser l'espace d'entrée et, en seconde couche, des modèles linéaires valables localement et sélectionnés par la carte SOM [16, 17, 43, 22] ou encore, à la place des modèles linéaires, des perceptrons multi-couches [43].

Les techniques à modèles multiples comprennent les approches faisant intervenir simultanément un modèle linéaire et un réseau multi-couches de perceptrons [2], les approches bi-directionnelles où l'on prédit aussi le passé [44], les mixtures d'experts [3], les approches où plusieurs modèles sont utilisés simultanément et auxquels on attribue une responsabilité pour la réponse finale [48] et les politiques de type "divide and conquer" où l'on utilise différents réseaux avec des fenêtres temporelles d'entrée et de sortie différentes [3]. La réponse finale est constituée d'un mélange entre les réponses des modèles multiples.

Catégorie	Techniques
Modèle unique	RBF FIR Rétro-propagation Régression
Modèles locaux	SOM + modèles linéaires locaux SOM + perceptrons multi-couches locaux
Modèles multiples	Approches bi-directionnelles Modèle linéaire + perceptrons multi-couches Mixture d'experts Multiples modèles avec responsabilité Divide and conquer

TAB. 1.1 – *Techniques de modélisation de séries temporelles.*

Toutes ces techniques posent trois problèmes avant d'être mises en oeuvre ainsi qu'un problème de validation. Le premier problème rencontré est de déterminer la taille de la fenêtre d'entrée [18, 9]. Beaucoup de travaux tentent de solutionner ce problème. Une fenêtre d'entrée trop petite ne permet pas de discerner correctement des vecteurs d'entrées semblables qui ont un résultat, en sortie, très différent. Une fenêtre trop grande rend les modèles plus complexes. Le second problème est lié à l'optimisation de la dimension des vecteurs d'entrées. Certaines approches visent à réduire la dimension de ces vecteurs en les transformant [20, 21, 19]. Ainsi, on diminue la complexité des modèles et le temps de calcul. Le troisième problème est de transformer les

entrées de manière à leur donner certaines propriétés [18, 25, 30]. Selon l'algorithme, l'apprentissage est facilité si les données fournies en entrées possèdent certaines caractéristiques (par exemple des caractéristiques statistiques). La convergence rapide des modèles dépend souvent du caractère indépendant des entrées. Ce problème est souvent abordé dans la littérature. Le principe général est de faire une transformation de l'espace d'entrée afin de rendre les entrées plus indépendantes. Ensuite, on prédit dans cet espace transformé et, enfin, on effectue une transformation inverse pour se ramener dans l'espace de sortie. Le dernier problème intervient après la mise en oeuvre du modèle. Il faut déterminer comment valider un modèle [23]. La plupart du temps les modèles sont validés par leurs performances sur le problème considéré.

1.5 Prédiction de séries temporelles issues d'un système dynamique théorique particulier

Cette approche possède un objectif d'identification du système dynamique sous-jacent (figure 1.3). Le tableau 1.2 synthétise les différentes méthodes existantes. Analysons une première méthode [31, 40, 38, 39]. Selon cette méthode, on considère une équation d'état particulière de la forme

$$y(n+1) = \mathcal{F}(\overrightarrow{u}(n), \overrightarrow{x}(n))$$

où $\mathcal{F}$ est la fonction du système dynamique sous-jacent à apprendre, $\overrightarrow{u}(n)$ est la fenêtre temporelle des entrées du système dynamique jusqu'à l'instant n et $\overrightarrow{x}(n)$ le vecteur d'état à l'instant n. Cette méthode considère que le système possède une sortie $y(n+1)$. La théorie nous apprend qu'un nombre suffisant d'observations des sorties du système dynamique observable correspond au vecteur d'état de ce même système. Nous pouvons donc dire que $\overrightarrow{x}(n)$ est équivalent à un certain vecteur $\overrightarrow{y}(n)$ constitué d'observations précédentes de la sortie du système :

$$\overrightarrow{y}(n) = \begin{pmatrix} y(n-td) \\ \vdots \\ y(n-2d) \\ y(n-1d) \\ y(n) \end{pmatrix}.$$

Nous avons, dès lors, de manière équivalente

$$y(n+1) = \mathcal{F}(\overrightarrow{u}(n), \overrightarrow{y}(n)).$$

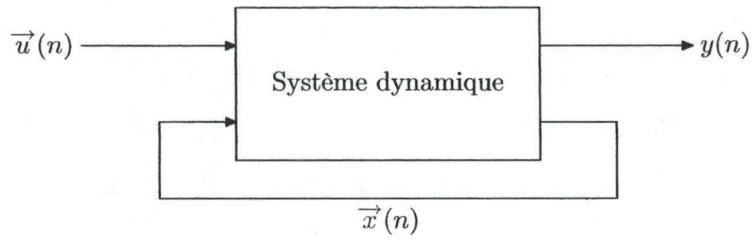


FIG. 1.3 – *Système dynamique.*

Cette méthode préconise de discrétiser l'espace d'entrée constitué par l'ensemble des vecteurs $[\overrightarrow{u}(n)\ \overrightarrow{y}(n)]$ à l'aide d'une carte auto-organisatrice de Kohonen (SOM) ou d'une variante en première couche et d'utiliser, en seconde couche, des modèles linéaires pour prédire localement la valeur de la sortie à l'instant $n+1$. L'union des modèles linéaires locaux constitue une fonction $\hat{\mathcal{F}}$ qui est une approximation de $\mathcal{F}$.

Considérons maintenant une autre méthode [13]. Dans cette approche, le système dynamique est décrit par les équations d'état suivantes :

$$\overrightarrow{x}(n+1) = \varphi(W_a \overrightarrow{x}(n) + W_b \overrightarrow{u}(n))$$

et

$$y(n) = C\overrightarrow{x}(n)$$

où $\varphi(.)$ est une fonction, W_a, W_b et C sont des matrices, $\overrightarrow{x}(n)$ le vecteur d'état (explicite) au temps n, $\overrightarrow{u}(n)$ le vecteur d'entrée au temps n et $y(n)$ la sortie au temps n.

Cette méthode utilise un réseau récurrent pour prédire $y(n)$ avec, en entrée, le vecteur $u(n)$. La fonction $\varphi(.)$ sera la fonction d'activation des neurones qui constituent le réseau récurrent. Le modèle suppose la connaissance de $\varphi(.)$. L'apprentissage de W_a et W_b se réalise avec l'algorithme de rétro-propagation à travers le temps.

Système dynamique considéré	Technique
$y(n+1) = \mathcal{F}(\overrightarrow{u}(n), \overrightarrow{y}(n))$	SOM + modèles linéaires locaux
$\overrightarrow{x}(n+1) = \varphi(W_a \overrightarrow{x}(n) + W_b \overrightarrow{u}(n))$ $y(n) = C\overrightarrow{x}(n)$	Réseaux récurrents de fonction d'activation $\varphi(.)$

TAB. 1.2 – *Techniques de modélisation de systèmes dynamiques.*

1.6 Prédiction de trajectoires balistiques

Le tableau 1.3 reprend un comparatif des méthodes abordées dans cette section. La première méthode dégagée se décompose en trois étapes (figure 1.4). La première

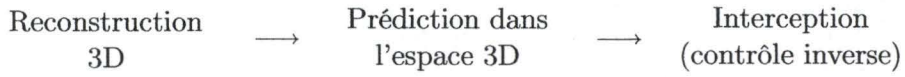


FIG. 1.4 – *Méthode des trois phases.*

consiste à reconstruire l'espace cartésien en trois dimensions. Cette reconstruction des coordonnées des objets peut être réalisée soit, par un logiciel, soit, le plus souvent, par du matériel spécialisé. La seconde étape de cette méthode consiste à prédire la trajectoire de la balle dans l'espace reconstruit. Les techniques utilisées sont le filtre de Kalman étendu [41], la régression polynômiale (de degré 2) [33, 32], un système mixte entre la régression polynômiale et le filtre de Kalman [33, 32] et l'utilisation de l'algorithme "recursive least square" (RLS) sur un modèle parabolique [15]. La troisième étape est un problème de contrôle inverse. Il s'agit du contrôle du bras robotique chargé de l'interception du projectile [33, 15]. Généralement, ce problème de contrôle inverse est traité directement dans l'espace reconstruit.

La deuxième méthode se compose de deux étapes (figure 1.5). La première demeure la reconstruction 3D. La seconde consiste à déplacer le bras du robot sans prédire la trajectoire mais en utilisant les observations de celle-ci. On cherche à calculer la

FIG. 1.5 – *Méthode des deux phases.*

trajectoire désirée de l'effecteur du bras du robot de telle sorte qu'elle intercepte la trajectoire de l'objet. Pour cela, on minimise l'erreur entre la trajectoire réelle de l'effecteur par rapport à sa trajectoire désirée mais aussi l'erreur entre la trajectoire désirée et la trajectoire de l'objet observée. Pour calculer la fonction de la trajectoire de l'effecteur, on utilise certaines propriétés dynamiques et kinématiques. En pratique, on utilise un polynôme de degré 4 pour cette fonction [28].

Les deux méthodes sont assez proches. La grande différence réside dans le caractère implicite ou non de prédiction de la trajectoire de la balle. Dans la seconde méthode, on ne prédit pas la trajectoire de la balle mais on en tient compte indirectement. En effet, pour pouvoir intercepter la trajectoire de la balle, il faut minimiser, de toute façon, la distance entre celle-ci et l'effecteur du robot.

Méthode	Caractéristique
"3 phases"	La prédiction de la trajectoire et l'interception du mobile sont considérées comme des problèmes distincts.
"2 phases"	La prédiction de la trajectoire et l'interception du mobile sont considérées comme un problème unique.

TAB. 1.3 – *Comparatif des méthodes d'interception de trajectoires balistiques.*

1.7 Critique

Le tableau 1.4 présente une synthèse de cette critique. Selon la vision proposée à la section 1.4, on ne peut pas vraiment parler d'identification de système. Théoriquement, cette approche est identique à l'approche 1.5. En effet, la fenêtre temporelle est équivalente au vecteur d'état. Cependant, les modèles obtenus demeurent difficilement interprétables à cause de la fenêtre temporelle. De plus, la taille et le délai de la fenêtre temporelle d'entrée doivent être trouvés, en partie, empiriquement sans jamais avoir la certitude que la taille et le délai trouvés soient les bons ou du moins les mieux adaptés pour résoudre le problème considéré. La validité du modèle sera établie sur base de sa performance.

La seconde approche dans la section 1.5 est plus intéressante que la première. En effet, elle se formule selon une représentation proche de la représentation d'état. Cela la rend plus explicite et interprétable. Néanmoins, cette seconde approche considère une fonction particulière du système, ce qui constitue une limitation. En opposition, la première approche ne pose aucune hypothèse sur la fonction du système mais fait appel à une fenêtre temporelle toujours déterminée de manière empirique. L'avantage des techniques "explicites" est la facilité d'interprétation du modèle obtenu. Cette approche semble donc plus adaptée à l'identification de système. La validité du modèle pourra être établie par d'autres moyens que la performance comme, par exemple, la valeur des coefficients après convergence.

Au niveau des approches pratiques du point 1.6, il faut remarquer que le cadre d'expérimentation est créé de façon à supprimer un maximum de difficultés pour le modèle de prédiction de trajectoire. Ce cadre est optimisé pour que les expériences fonctionnent bien. Ces optimisations limitent, évidemment, la portée des résultats.

Approche	Points positifs	Points négatifs
Séries temporelles (1.4)	– Aucune connaissance du système dynamique nécessaire	– Validation par les performances – Fenêtre temporelle déterminée empiriquement – Pas d'identification de système – Modèles difficilement interprétables
Systèmes dynamiques (1.5)	– Identification du système dynamique – Interprétation des modèles possibles – Validation possible par les coefficients	– Nécessite la plupart du temps des connaissances sur le système sous-jacent
"Catching ball" (1.6)	– Approche de type ingénierie	– Conditions d'expérimentation optimisées – Pas d'identification de système dynamique

TAB. 1.4 – *Avantages et désavantages des approches.*

Remarquons encore que la plupart de ces approches font appel à un matériel spécialisé qui permet un échantillonnage très précis et un traitement des données très rapide. Ainsi, dans ces approches, on utilise des caméras parallèles fortement écartées, ce qui permet de reconstruire aisément les coordonnées cartésiennes de la balle. De plus, l'écartement important des caméras diminue le bruit dû à la pixélisation lors du calcul de la coordonnée de profondeur. Certaines approches vont même jusqu'à placer le plan de la trajectoire balistique le plus parallèlement possible par rapport au plan des caméras, ce qui annule fortement l'effet de la perspective. Les résultats vidéos de certaines approches disponibles sur internet donnent l'illusion d'un algorithme puissant alors que les conditions d'expérimentations sont créées de manière à réussir une manipulation particulière.

1.8 Problématique en rapport avec la littérature

Notre problématique va se démarquer de celles récemment traitées dans la littérature et ce, de plusieurs manières.

Premièrement, d'un point de vue théorique, nous n'avons pas trouvé d'articles traitant de la prédiction de trajectoires balistiques qui utilisent des approches neuromimétiques dans un but d'identification de système. Tout au plus, certaines approches sont utilisées en substitution de régressions traditionnelles. Nous tenterons, dans la mesure du possible, de rencontrer un objectif d'identification de système. Cependant, si au cours de nos recherches, cet objectif semble inaccessible pour certaines raisons, nous nous résignerons à des approches plus proches de l'ingénierie.

Deuxièmement, d'un point de vue pratique, nous ne disposons pas de matériel spécialisé pour observer la scène. Le seul matériel disponible est constitué de webcams tout à fait standard. Cet élément rajoute une forte contrainte par rapport à la littérature traitant de la prédiction de trajectoires balistiques, à savoir que nous ne bénéficierons que d'un taux d'échantillonnage très faible.

Troisièmement, contrairement aux articles de la littérature, nous ne fixerons aucune contrainte sur la scène ou sur le plan de la trajectoire (exemple: le plan de trajectoire parallèle au plan des images). Les seules contraintes que nous accepterons, sont celles liées à la plate-forme robotique, les contraintes de bon sens et les contraintes que nous ajouterons au départ pour élaborer notre raisonnement mais que nous tenterons de lever par la suite.

1.9 Conclusion

Ce mémoire va constituer une approche originale du problème de prédiction de trajectoires balistiques. En effet, cette prédiction se réalisera dans les "espaces images" des caméras en utilisant un matériel non spécialisé. De plus, un minimum, voire aucune connaissance de la scène, ne devra être exploité dans une solution finale. Nous nous fixerons un objectif intermédiaire d'identification de système qui pourra être, éventuellement, abandonné pour des raisons pratiques. Ce mémoire se veut donc en marge de ce qui existe actuellement dans la littérature. En atteignant les objectifs cités ci-dessus, il réalisera un apport important d'éléments nouveaux en supprimant un maximum de contraintes liées aux conditions d'expérimentations et en utilisant un matériel non spécialisé.

Chapitre 2

Réseaux de neurones

2.1 Introduction

Nous introduisons dans ce chapitre les réseaux de neurones artificiels. Ce chapitre n'a pas pour ambition de dresser l'état de l'art des réseaux de neurones. L'objectif est d'introduire les éléments théoriques utiles à une bonne compréhension des chapitres suivants.

Nous commencerons par présenter un bref historique qui retracera le contexte dans lequel les réseaux de neurones ont émergé. Ensuite, nous partirons de l'intuition du fonctionnement biologique du neurone pour en déduire une modélisation de celui-ci. Nous introduirons alors l'élément adaline et nous nous attarderons sur ses méthodes d'apprentissage. Un petit résumé des principales architectures de réseaux de neurones sera ensuite développé. Finalement, nous clôturerons par une explication des différentes caractéristiques des réseaux de neurones.

2.2 Contexte historique

La modélisation de toute tâche automatisée repose sur la décomposition et la description de cette tâche en termes de règles. Ce principe constitue la principale limitation de l'informatique moderne. En d'autres termes, tout problème qui est facilement décrit et analysé, est facilement programmé. Dès lors, il devient aisé de le résoudre avec les moyens informatiques actuels. Mais ce qui est difficile à décrire en termes de règles est aussi malaisé à solutionner simplement à cause de l'inadaptation des techniques de l'informatique.

L'idée de départ des réseaux de neurones artificiels est la suivante : si l'être humain peut facilement maîtriser des problèmes qui semblent si complexes pour l'informatique traditionnelle, la solution serait, peut-être, de construire des machines qui "imitent" le fonctionnement du cerveau humain.

Des chercheurs ont donc essayé de comprendre, dès le milieu du siècle passé, le fonctionnement du cerveau. De manière plus réaliste, ils ont observé comment s'organisent et communiquent quelques neurones entre eux, en vue d'apporter des améliorations aux outils informatiques de demain à l'aide des capacités de traitement des neurones.

D'où viennent les formidables capacités du cerveau telles que la coordination

sensori-motrice, la généralisation d'un apprentissage, la reconnaissance de la parole? Le parallélisme massif des neurones et l'adaptation d'un grand nombre de paramètres peuvent constituer une grande source d'inspiration. Se basant sur ces deux concepts, des chercheurs ont essayé de modéliser le comportement du neurone.

Il faut remarquer que certains types de réseaux de neurones sont loin de toute plausibilité biologique. En effet, ils ont évolué comme outils indépendamment des connaissances biologiques actuelles. Or, il existe une différence entre, essayer de modéliser le fonctionnement des neurones, et modéliser leur capacité de traitement, malgré une inspiration biologique sous-jacente. Dès lors, le but est de créer des outils performants qui ne cherchent plus à avoir de pouvoir explicatif sur le fonctionnement du cerveau.

Le principal intérêt des recherches dans les réseaux de neurones se trouve dans son caractère inter-disciplinaire. On utilise les idées offertes par le monde biologique, les outils développés par les mathématiques et l'informatique pour construire de nouveaux outils applicables à de nombreux domaines [42].

La recherche dans le domaine des réseaux de neurones s'est enrichie durant trois périodes de développement intense. Les premiers travaux ont été réalisés en 1943 par Warren McCulloch, psychologue et neuro-anatomiste, et par Walter Pitts, un mathématicien prodige. Dans leur texte, "A Logical Calculus Immanent in Nervous Activity", ceux-ci proposaient que les cellules nerveuses jouent le rôle d'éléments logiques, une évaluation booléenne de fonctions logiques. Le neurone formel est né dans ce contexte. Avec un nombre suffisant de ces éléments opérant simultanément, et des connections synaptiques correctement paramétrées, McCulloch et Pitts montrèrent qu'un réseau ainsi constitué pourrait, en principe, calculer toute fonction logique calculable [13]. Ces résultats significatifs fondèrent la discipline des réseaux de neurones.

Le développement majeur suivant, durant cette première période, vint en 1949 avec la publication du livre "The Organization of Behavior" de Donald Hebb, dans lequel une formulation du mécanisme d'apprentissage, sous forme d'une règle de modification des connexions synaptiques, est présentée pour la première fois.

La deuxième période de développement intense commença par l'introduction d'une nouvelle approche du problème de reconnaissance de "patterns" par Frank Rosenblatt en 1958. Son travail sur le perceptron déboucha sur une méthode originale d'apprentissage supervisé. Le couronnement de son travail est appelé "le théorème de convergence du perceptron" qui est considéré comme la première preuve des avancées de Rosenblatt.

En 1960, Bernard Widrow et Ted Hoff ont introduit l'algorithme "least mean-square" (LMS) et l'utilisèrent pour formuler l'adaline ("adaptive linear element"). La différence entre le perceptron et l'adaline se situe non seulement dans le caractère

linéaire, mais aussi, et essentiellement, dans la procédure d'apprentissage utilisée.

Durant les années 60, les réseaux de neurones semblaient avoir des capacités très prometteuses. Mais en 1969, Marvin Minsky et Seymour Papert montrèrent qu'il existe des limites fondamentales sur ce que les perceptrons mono-couche peuvent calculer. De plus, dans une brève section sur les perceptrons multi-couches, ils affirmèrent qu'il n'y avait pas de raison de penser que ces limitations puissent être outrepassées par la version multi-couches.

Les résultats de Minsky et Papert découragèrent donc d'entreprendre et de subventionner de telles recherches et de fait, la plupart des chercheurs désertèrent le domaine dans les années 70. De plus, l'absence de technologie permettant l'expérimentation, constitua un frein supplémentaire.

Cependant, au cours de cette période, des recherches importantes en conséquences furent entreprises sur les cartes auto-organisatrices utilisant l'apprentissage compétitif. Avec les résultats des travaux de Christoph von der Malsburg (1973) et de David Willshaw sur l'auto-organisation, Teuvo Kohonen popularisa et généralisa les cartes auto-organisatrices (SOM, Self-Organizing Maps). Ce type de réseaux de neurones a l'avantage de pouvoir s'appliquer à une grande quantité de problèmes, tout en restant plausible biologiquement [35].

La troisième période débuta dans les années 80 lorsqu'émergea un regain d'intérêt pour les réseaux de neurones. Ce renouveau était principalement dû à la vulgarisation de l'algorithme de "back-propagation" et des réseaux récurrents de Hopfield.

C'est en 1982 que John Hopfield démontra l'utilité des réseaux complètement connectés dans la compréhension et la modélisation des processus de la mémoire.

En 1986, le développement de l'algorithme de "back-propagation" fut rapporté par David Rumelhart, Geoffrey Hinton et Ronald Williams. La même année, le célèbre livre en deux volumes "Parallel Distributed Processing: Explorations in the Microstructures of Cognition" écrit par Rumbelhart et James McClelland, fut publié. Celui-ci eut une importante influence dans l'utilisation de la "back-propagation" dans les réseaux multi-couches.

Enfin, beaucoup d'autres techniques et algorithmes ont été inventés et développés lors de cette moitié de siècle. Nous pouvons citer les mémoires associatives, les filtres d'apprentissage non-linéaire et la théorie de résonance adaptative (ART), ...

2.3 Modélisation d'un neurone biologique

2.3.1 Introduction

Le système nerveux humain peut être grossièrement schématisé par un système composé de trois parties: les récepteurs, le cerveau et les effecteurs. En simplifiant un maximum le fonctionnement du cerveau humain, on peut le considérer comme un ensemble de neurones inter-connectés.

Dès lors, selon le modèle énoncé ci-dessus, un stimulus est identifié par des récepteurs qui le transforment en signal électrique. Celui-ci peut alors être traité par le réseau neuronal qui constitue notre cerveau. S'il y a lieu, des effecteurs sont sollicités et transforment la réponse électrique du réseau en une réponse discernable par les systèmes de sortie.

Ce cheminement est présenté à la figure 2.1. La présence de flèches allant de la gauche vers la droite, dans cette figure, traduisent un "feed-back" d'information. Ce "feed-back" d'information est difficile à cerner. La biologie tente toujours de comprendre comment les récepteurs et les effecteurs interagissent avec notre cerveau.

Une tentative d'explication serait de considérer que, lorsque les effecteurs ont traité l'information qui leur a été transmise, ils envoient, à leur tour, des informations sur ce traitement. On peut admettre que ces informations remontent dans le système nerveux mais pas nécessairement jusqu'au cerveau.

Quant aux récepteurs, illustrons la flèche allant du réseau neuronal vers les récepteurs à l'aide d'un exemple. Lors d'une conversation dans un endroit où il y a plusieurs personnes qui parlent, nous arrivons, grâce à la concentration, à discerner une seule voix. Si notre attention se relâche, nous ne discernons plus rien. Cet exemple tente d'illustrer la dépendance des récepteurs par rapport au cerveau.

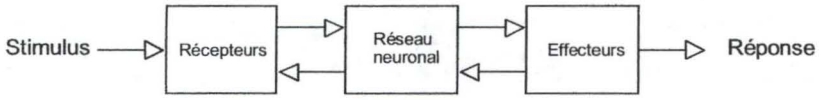


FIG. 2.1 – *Représentation du système nerveux.*

Une fois que le stimulus a été converti par les récepteurs en signal électrique intelligible par les neurones du cerveau, un ou plusieurs neurones sont stimulés.

La figure 2.2 montre un neurone typique avec ses différentes parties. Les dendrites constituent la zone réceptive du neurone. Le neurone produit une réponse en fonction

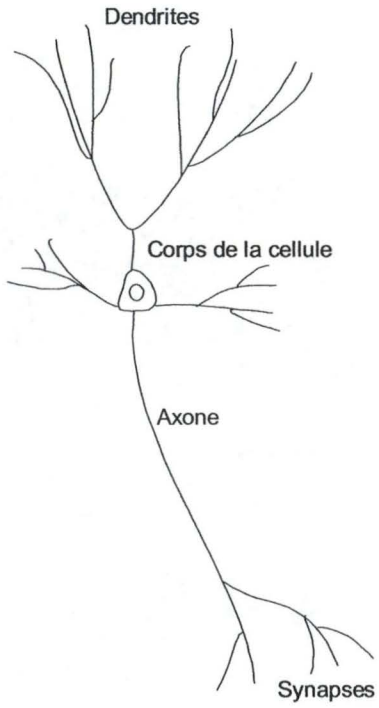


FIG. 2.2 – *Neurone biologique.*

des stimulations excitatrices ou inhibitrices qu'il a reçues au travers de ses dendrites. La réponse est acheminée au travers de l'axone vers les synapses. Celles-ci constituent les liaisons inter-neuronales. Elles peuvent être excitatrices ou inhibitrices, mais elles ne peuvent exercer qu'un des deux effets.

Remarquons que les synapses chimiques opèrent une conversion du signal électrique en signal chimique au niveau pré-synaptique et le reconvertissent au niveau post-synaptique.

2.3.2 Formalisation

De cette connaissance du neurone biologique, on peut déduire un certain nombre d'éléments qui vont permettre de le modéliser de manière mathématique :

1. le neurone va collecter les signaux excitateurs et inhibiteurs qui proviennent des synapses d'autres neurones qui lui sont connectés, au travers de ses dendrites.
2. il va les fusionner en un potentiel d'excitation ou d'inhibition.
3. le neurone va donc réagir face à ce potentiel avec plus ou moins de force. Cette réaction constitue sa réponse qui sera acheminée au travers de l'axone vers les synapses excitatrices ou inhibitrices d'autres neurones.

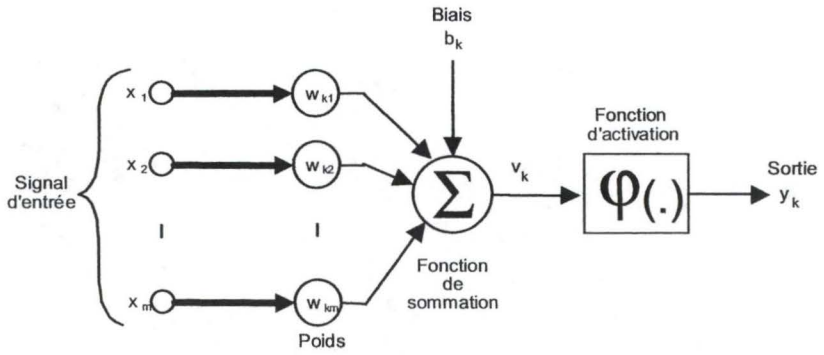


FIG. 2.3 – *Neurone formel.*

On peut donc modéliser un neurone biologique comme suit [13] :

1. une synapse sera caractérisée par deux éléments : son entrée x_j et son poids w_{kj} qui constitue son caractère excitateur ou inhibiteur. L'indice k représente le neurone en question et l'indice j, l'entrée concernée. Pour une synapse j d'un neurone k, il conviendra de multiplier le poids w_{kj} par l'entrée x_j pour connaître le signal $w_{kj}x_j$ résultant.
2. une sommation servira à additionner les différents signaux résultant des synapses pour obtenir la combinaison d'entrées u_k. On y adjoindra un biais b_k pour aboutir à v_k.
3. la combinaison des entrées et du biais v_k sera ensuite transformée au moyen d'une fonction d'activation, notée $\varphi(.)$, qui servira à limiter l'amplitude de la réponse du neurone. Cette transformation sera notée y_k.

Dès lors, un neurone k est caractérisé par deux équations [13] :

$$u_k = \sum_{j=1}^{m} w_{ki}x_j, \tag{2.1}$$

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

où $x_1, x_2, \ldots, x_m$ sont les signaux des entrées, $w_{k1}, w_{k2}, \ldots, w_{km}$ sont les poids synaptiques du neurone, u_k est la combinaison des entrées, b_k est le biais, φ est la fonction d'activation et y_k est la sortie du neurone k. On peut représenter la modélisation du neurone par la figure 2.3.

Posons

$$v_k = u_k + b_k. \tag{2.3}$$

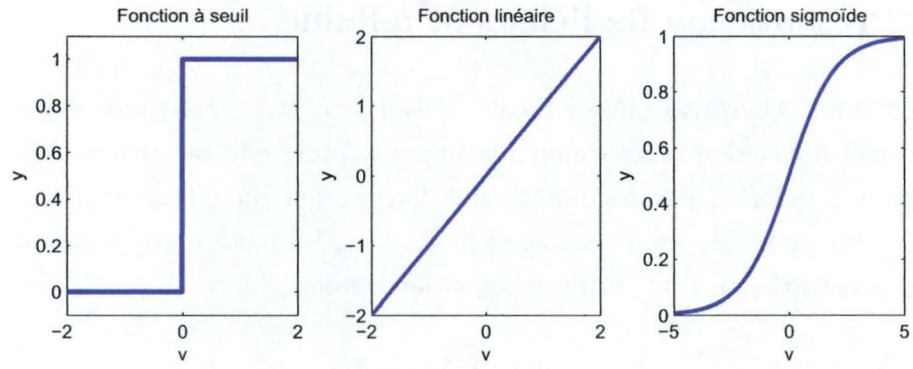


FIG. 2.4 – *Fonctions d'activation.*

Imposons que $x_0 = 1$ et $w_{k0} = b_k$, notre système d'équations prend une forme généralisée :

$$v_k = \sum_{j=0}^{m} w_{kj} x_j, \tag{2.4}$$

$$y_k = \varphi(v_k). \tag{2.5}$$

Passons en notation matricielle, avec $\vec{w_k}$ le vecteur $[w_{k1}, w_{k2}, \ldots, w_{km}]$ et $\vec{x}$ le vecteur $[x_1, x_2, \ldots, x_m]^T$

$$u_k = \vec{w_k}\vec{x}, \tag{2.6}$$

$$y_k = \varphi(u_k + b_k). \tag{2.7}$$

Les trois fonctions d'activation $\varphi(.)$ les plus utilisées sont: la fonction à seuil, la fonction linéaire et la fonction de type sigmoïde. Ces trois fonctions sont représentées à la figure 2.4.

En pratique, on utilisera surtout des fonctions de type sigmoïde, car elles présentent de nombreux avantages, notamment elles généralisent la fonction à seuil. L'expression d'une fonction de type sigmoïde peut être [13]:

$$\varphi(v) = \frac{1 - e^{-av}}{1 + e^{-av}} = tanh(\frac{av}{2}) \quad , \quad \frac{d\varphi}{dv} = \frac{a}{2}[1 - \varphi^2(v)] \tag{2.8}$$

ou encore :

$$\varphi(v) = \frac{1}{1 + e^{-av}} \quad , \quad \frac{d\varphi}{dv} = a\varphi(v)[1 - \varphi(v)] \tag{2.9}$$

où a est le paramètre de pente de la fonction. Pour être exacte, la pente doit être égale à $a/4$ à l'origine de la fonction.

2.4 Présentation de l'élément adaline

L'adaline ("ADAptive LINear Element")[46] peut être considéré comme un neurone formel à fonction d'activation linéaire. L'adaline effectue une sommation de ses entrées pondérées par ses poids, sans l'utilisation d'un biais explicite. Soient $x_1,x_2,\ldots,x_m$ les entrées de l'adaline, $w_1,w_2,\ldots,w_m$ les poids correspondant aux entrées et y sa sortie, on peut écrire son équation comme:

$$y = \sum_{j=1}^{m} w_j x_j. \tag{2.10}$$

Si on préfère la notation matricielle plus intuitive, définissons $\vec{w} = [w_1,w_2,\ldots,w_m]^T$ et $\vec{x} = [x_1,x_2,\ldots,x_m]^T$ et l'équation de l'adaline s'écrit en utilisant le produit scalaire de deux vecteurs :

$$y = \vec{w}^T \vec{x}. \tag{2.11}$$

D'un manière géométrique, pour une sortie y fixe, l'équation 2.11 représente un hyper-plan défini par le vecteur $\vec{w}$ dans l'espace des poids à m dimensions.

2.5 Apprentissage de l'adaline

2.5.1 Introduction

L'adaline réalise son apprentissage, c'est-à-dire l'adaptation de ses poids, grâce à une règle d'adaptation. Il existe plusieurs règles d'adaptation. Nous pouvons citer le Least Mean Square (LMS), le α-LMS, aussi appelé Normalised Least Mean Square (NLMS), qui en est une version normée ou encore le Recursive Least Square (RLS), une version récursive de l'algorithme LMS [46, 10, 13, 26].

Le LMS a la particularité d'être l'un des éléments le plus utilisé depuis ces 40 dernières années dans le monde du traitement du signal adaptatif (Adaptive Signal Processing) car il est simple, robuste et permet une adaptation en ligne. C'est aussi pourquoi nous avons décidé de l'utiliser dans nos différentes approches.

Les méthodes d'apprentissage que nous décrirons sont du type supervisé. Contrairement aux méthodes non-supervisées, l'algorithme a besoin de connaître, pendant la phase d'apprentissage, une réponse désirée pour chaque entrée présentée au réseau. Cette correspondance permet à l'algorithme d'adapter les poids du réseau pour que celui-ci puisse généraliser son apprentissage par la suite.

Un ensemble de paires, réponse désirée - entrée correspondante, constitue alors l'ensemble d'apprentissage. Dans la suite, nous désignerons l'ensemble d'apprentissage

par la lettre $\mathcal{A}$.

L'utilisation d'un réseau avec une méthode d'apprentissage supervisée se réalise donc en deux phases : une phase d'apprentissage se servant de l'ensemble d'apprentissage, où les poids du réseau sont adaptés pour correspondre au modèle souhaité, et une phase d'utilisation, où les poids du réseau ne sont plus modifiés.

2.5.2 Erreur et performance

Dans le but d'adapter les poids de l'adaline, une sortie désirée du réseau doit être donnée. Considérons un ensemble de vecteurs d'entrées $\vec{x}_1, \ldots, \vec{x}_L$ et l'ensemble des sorties désirées correspondantes $d_1, \ldots, d_L$. Nous définirons l'erreur instantanée du réseau comme

$$\epsilon_k = d_k - y_k \tag{2.12}$$

où d_k est la sortie désirée du réseau et y_k la sortie effective de celui-ci au temps k. Cette erreur donne une indication de la performance du modèle au temps k. C'est pourquoi, nous pouvons utiliser cette indication de performance pour corriger le vecteur des poids $\vec{w}_k$.

Les méthodes d'apprentissage instantané, qui, comme le LMS, ne s'adaptent qu'en utilisant l'entrée et la sortie désirée à l'instant k, peuvent être considérées comme des approximations de règles utilisant l'ensemble des données d'entraînement [5]. Ces règles minimisent alors une fonction de performance prédéfinie. Il en existe plusieurs et nous utiliserons dans la suite du développement, la fonction de performance MSE (Mean Squared Error Performance):

$$J = E(\epsilon_k^2) = \frac{1}{L}\sum_{k=1}^{L}(d_k - y_k)^2. \tag{2.13}$$

En effet, un vecteur de poids optimal peut être trouvé à partir d'un ensemble d'apprentissage statique fini. Mais cet apprentissage hors ligne coûte algorithmiquement plus qu'un apprentissage en ligne. Ce dernier est mieux approprié pour un apprentissage en temps réel où l'on découvre au fur et à mesure l'ensemble d'apprentissage.

La fonction de performance MSE peut être interprétée comme une hyper-surface dans un espace à $m+1$ dimension, où m est le nombre de poids. On peut donc, en évaluant le MSE pour un vecteur de poids $\vec{w}$ donné, construire la surface de performance liée à la fonction [5] :

$$\epsilon_k = d_k - y_k = d_k - \vec{x}_k^T\vec{w}, \tag{2.14}$$

$$\epsilon_k^2 = d_k^2 + \vec{w}^T\vec{x}_k\vec{x}_k^T\vec{w} - 2d_k\vec{x}_k^T\vec{w}. \tag{2.15}$$

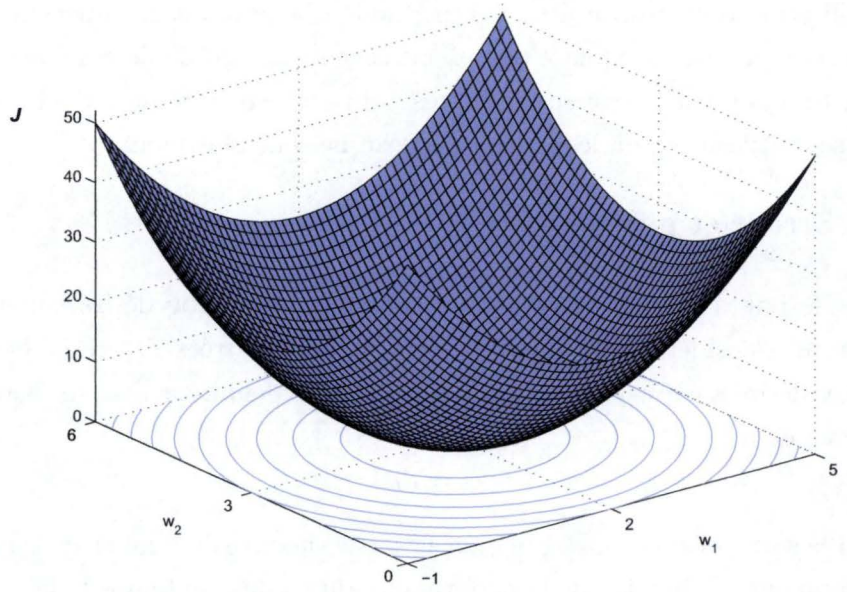


FIG. 2.5 – *Une surface de performance MSE typique dans l'espace des poids à deux dimensions. Le vecteur de poids optimal est* $\hat{\vec{w}} = (4,3)^T$ *[5].*

Avec la définition de la fonction de performance à l'équation 2.47 :

$$J = E(d_k^2) + \vec{w}^T E(\vec{x}_k \vec{x}_k^T)\vec{w} - 2E(d_k \vec{x}_k^T)\vec{w}, \tag{2.16}$$

si nous définissons la matrice d'autocorrélation R comme :

$$R = E(\vec{x}_k \vec{x}_k^T) \tag{2.17}$$

et le vecteur d'intercorrélation $\vec{p}$ comme :

$$\vec{p} = E(d_k \vec{x}_k), \tag{2.18}$$

nous obtenons la simplification suivante :

$$J = E(d_k^2) + \vec{w}^T R\vec{w} - 2\vec{p}^T\vec{w}. \tag{2.19}$$

Il est important de remarquer qu'il s'agit d'une forme quadratique en terme du vecteur de poids $\vec{w}$. Il ne peut exister d'extremum local, au plus une infinité d'extremums globaux. De plus, si la matrice d'autocorrélation est non singulière, c'est-à-dire si les entrées sont linéairement indépendantes, il n'existe qu'un et un seul extremum global. Une surface de performance MSE typique est visible à la figure 2.5.

2.5.3 Solution directe

Pour un ensemble d'entraînement complet $\{\vec{x}_k,d_k\}_{k=1}^{L}$, le vecteur de poids qui minimise la fonction de performance MSE peut être directement calculé en utilisant la pseudo-inverse de Moore-Penrose. Cette technique correspond, en fait, à la régression linéaire. En effet, pour un tel ensemble, l'équation 2.16 s'écrit :

$$J = \vec{d}^T\vec{d} + \vec{w}^T A^T A\vec{w} - 2A^T\vec{d}\vec{w} \tag{2.20}$$

où $\vec{d}$ est composé des sorties désirées d_k, A est une matrice de dimension $(L \times m)$, où la ligne k est composée du vecteur d'entrée $\vec{x}_k$.

En posant le gradient égal à zéro, nous obtenons :

$$\nabla_{\vec{w}} J = \frac{\partial J_k}{\partial \vec{w}} = 2A^T A\vec{w} - 2A^T\vec{d} = 0, \tag{2.21}$$

$$\hat{\vec{w}} = (A^T A)^{-1} A^T\vec{d} \tag{2.22}$$

où $\hat{\vec{w}}$ est le vecteur optimal et $(A^T A)^{-1} A^T\vec{d}$ est appelé la pseudo-inverse de Moore-Penrose.

Nous retrouvons la matrice d'autocorrélation R :

$$R = \frac{1}{L} A^T A \tag{2.23}$$

et le vecteur d'intercorrélation $\vec{p}$:

$$\vec{p} = \frac{1}{L} A^T\vec{d}. \tag{2.24}$$

Si la matrice d'autocorrélation n'est pas singulière, cela nous amène à l'équation :

$$\hat{\vec{w}} = R^{-1}\vec{p}. \tag{2.25}$$

2.5.4 Descente de gradient instantané

D'une manière simple, intuitive et géométrique, étant donné une position quelconque sur la surface de performance, c'est-à-dire un vecteur de poids initial aléatoire, le principe de la descente de gradient est de descendre la surface de performance jusqu'au point le plus bas, c'est-à-dire là où le gradient s'annule.

Cela est réalisé par le calcul du gradient de la fonction de performance au point courant. De cette façon, vu qu'il n'existe pas d'optimum local, nous procédons à la recherche du chemin nous emmenant, dans le cas où celle-ci est unique, à la solution optimale.

Les méthodes d'apprentissage instantané réalisent des approximations de l'algorithme de descente du gradient original. Celles-ci utilisent alors une estimation instantanée de la fonction de performance.

Soit une estimation instantanée au temps k de la fonction de performance définie à l'équation 2.47:

$$J_k = \epsilon_k^2 = (d_k - y_k)^2 = (d_k - \vec{x}_k^T \vec{w})^2, \tag{2.26}$$

$$J_k = d_k^2 - 2d_k \vec{x}_k^T \vec{w} + (\vec{x}_k^T \vec{w})^2. \tag{2.27}$$

Estimons son gradient :

$$\nabla_{\vec{w}} J = (-2d_k + 2\vec{x}_k^T \vec{w})\vec{x}_k, \tag{2.28}$$

$$\nabla_{\vec{w}} J = -2\epsilon_k \vec{x}_k. \tag{2.29}$$

2.5.5 L'algorithme Least Mean Square (LMS)

De manière à progresser vers la solution optimale, nous devons adapter le vecteur des poids d'une certaine proportion dans le sens opposé du gradient instantané. Nous choisissons donc de mettre à jour le vecteur des poids de :

$$\triangle \vec{w}_k = \delta \epsilon_k \vec{x}_k \tag{2.30}$$

où $\triangle \vec{w}_k = \vec{w}_{k+1} - \vec{w}_k$ et δ étant un taux d'apprentissage.

Cette dernière règle constitue le corps de la méthode de descente du gradient instantané Least Mean Square (LMS). Elle est de la forme :

Poids de mise à jour = scalaire × vecteur d'entrée.

Nous voyons que le chemin de recherche est parallèle au vecteur d'entrée $\vec{x}_k$ à l'instant k et que la taille du pas d'avancement est égale au taux d'apprentissage multiplié par l'erreur de sortie instantanée.

L'interprétation géométrique est, en effet, très simple, comme montré à la figure 2.6. Dans l'espace des poids, le vecteur de changement des poids $\triangle \vec{w}_k$, ajouté au vecteur des poids $\vec{w}_k$, est parallèle au vecteur d'entrée $\vec{x}_k$ à l'instant k. Seuls les poids qui contribuent à la sortie sont ainsi adaptés. Par contre, si le vecteur d'entrée contient une composante presque nulle, la composante correspondante du vecteur de poids ne sera presque pas adaptée.

Une surface de performance instantanée générée par un entraînement est montrée à la figure 2.7. A chaque paire d'entraînement correspond une équation d'un hyper-plan

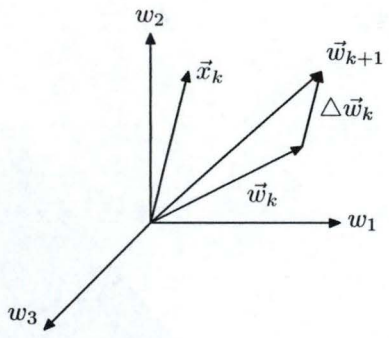


FIG. 2.6 – *Interprétation géométrique des règles d'apprentissage du type Least Mean Square. Ceci est un exemple d'un adaline à trois entrées et donc trois poids. Nous voyons que, dans l'espace des poids, le vecteur de changement des poids $\triangle\vec{w}_k$, ajouté au vecteur des poids $\vec{w}_k$, est parallèle au vecteur d'entrée $\vec{x}_k$ à l'instant k [45].*

que les poids doivent satisfaire pour enregistrer exactement l'information:

$$\vec{x}_k^T \vec{w} = d_k. \tag{2.31}$$

Lorsqu'il n'existe qu'un seul minimum global, ces surfaces se coupent en un point unique qui correspond à l'optimum global de la fonction de performance MSE.

Plusieurs preuves de convergence du Least Mean Square existent dans la littérature, cfr. [46, 13, 5]. En résumé, la plus forte condition de convergence et de stabilité est :

$$0 < \delta < \frac{2}{\lambda_{max}} \tag{2.32}$$

où λ_{max} est la plus grande valeur propre de la matrice d'autocorrélation R.

De l'équation 2.17, la matrice instantanée d'autocorrélation R_k associée à la fonction J_k prend la forme :

$$R_k = \vec{x}\vec{x}^T. \tag{2.33}$$

C'est une matrice de dimension $m \times m$ qui ne possède qu'une valeur propre non nulle $|\vec{x}_k|^2$. Le vecteur propre correspondant étant $\vec{x}_k$ [5], la condition de stabilité devient alors :

$$0 < \delta < \frac{2}{|\vec{x}_k|^2}. \tag{2.34}$$

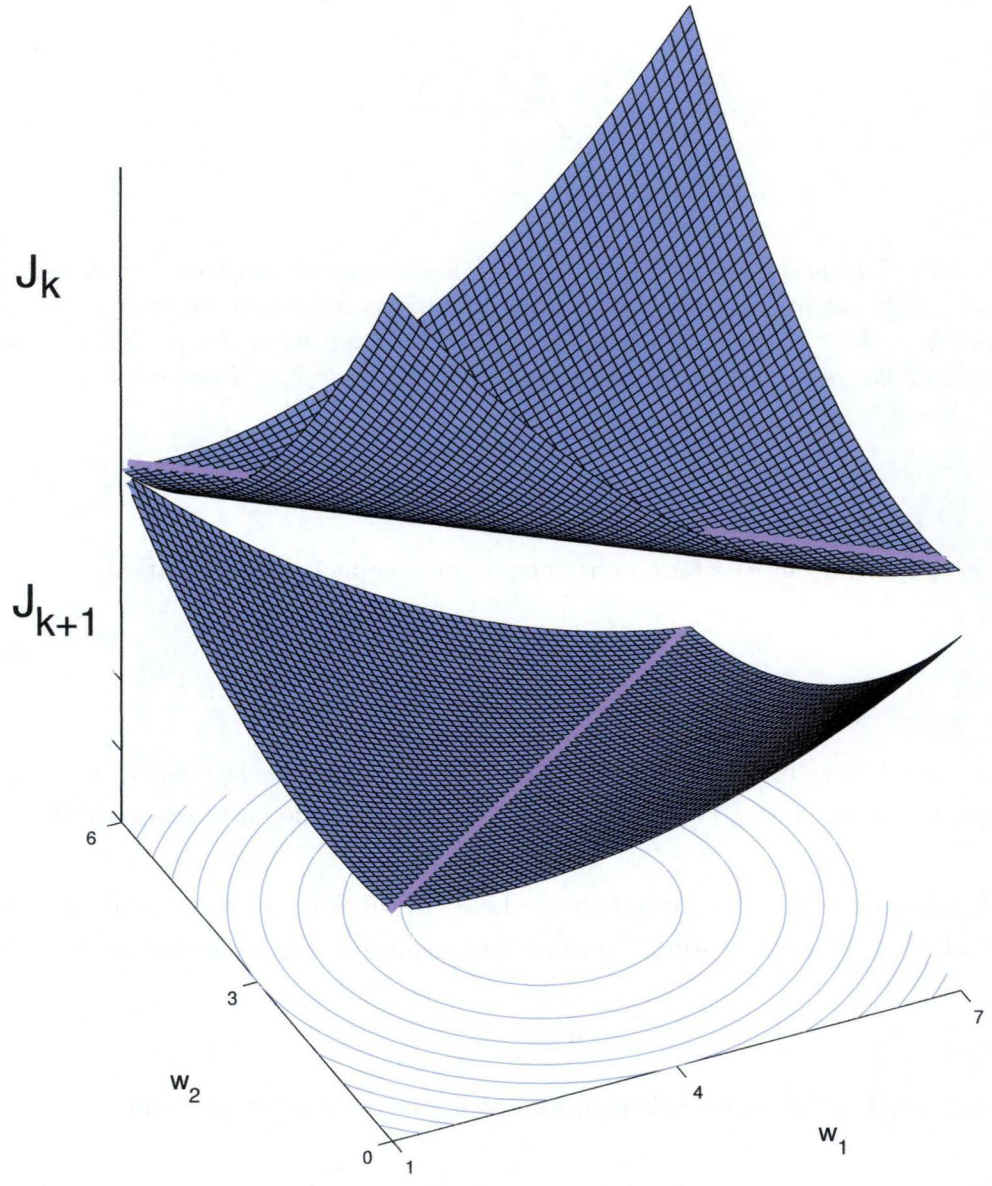


FIG. 2.7 – *Représentation de deux surfaces de performance instantanée aux instants k et $k+1$. A chaque paire d'entraînement correspond l'équation d'un hyper-plan $\vec{x}_k^T \vec{w} = d_k$ que les poids doivent satisfaire pour enregistrer exactement l'information. Les deux droites en clair correspondent donc à l'information apportée par les deux paires d'entraînement $\{\vec{x}_k, d_k\}$ et $\{\vec{x}_{k+1}, d_{k+1}\}$. Elles correspondent au minimum global de chacune de ces surfaces. La projection de ces droites se coupe au centre des contours projetés de la surface de performance MSE attendue. [5].*

2.5.6 L'algorithme alpha-LMS (α-LMS ou NLMS)

Après l'adaptation du vecteur de poids, celui-ci devrait, pour enregistrer exactement l'information, satisfaire l'équation :

$$d_k = \vec{x}_k^T \vec{w}_{k+1}. \tag{2.35}$$

Or, nous pouvons reformuler l'équation 2.30 comme :

$$\triangle \vec{w}_k = c_k \vec{x}_k. \tag{2.36}$$

Ceci nous amène à :

$$\begin{aligned} d_k &= \vec{x}_k^T(\vec{w}_k + c_k\vec{x}_k) && (2.37)\\ &= \vec{x}_k^T\vec{w}_k + c_k\vec{x}_k^T\vec{x}_k && (2.38)\\ &= y_k + c_k\vec{x}_k^T\vec{x}_k, && (2.39)\\ \epsilon_k &= c_k\vec{x}_k^T\vec{x}_k. && (2.40) \end{aligned}$$

Nous obtenons :

$$c_k = \frac{\epsilon_k}{\vec{x}_k^T\vec{x}_k} = \frac{\epsilon_k}{|\vec{x}_k|^2}. \tag{2.41}$$

Il en résulte la règle d'apprentissage suivante :

$$\triangle \vec{w}_k = \alpha \frac{\epsilon_k \vec{x}_k}{|\vec{x}_k|^2} \tag{2.42}$$

où α est le taux d'apprentissage.

Cette règle est appelée "Normalised Least Mean Square" (NLMS), ou encore, alpha-LMS [5, 13, 45, 46]. Elle est toujours de la forme *Poids de mise à jour = scalaire × vecteur d'entrée*, excepté que l'amplitude du vecteur de mise à jour est normalisée par l'amplitude du vecteur d'entrée. La direction du vecteur de mise à jour est toujours identique au LMS mais la taille du pas est différente.

On peut démontrer que l'erreur est réduite pour chaque vecteur $\vec{x}_k$. La variation d'erreur vaut pour une itération k:

$$\Delta\epsilon_k = \Delta(d_k - \vec{w}_k\vec{x}_k) = -\vec{x}_k^T\Delta\vec{w}_k^T. \tag{2.43}$$

En combinant les équations 2.43 et 2.42, nous avons

$$\Delta\epsilon_k = -\alpha\frac{\epsilon_k\vec{x}_k^T\vec{x}_k}{|\vec{x}_k|^2} = -\alpha\epsilon_k. \tag{2.44}$$

Finalement, nous obtenons :

$$\Delta\epsilon_k = \epsilon_{k+1} - \epsilon_k = -\alpha\epsilon_k, \tag{2.45}$$

$$\epsilon_{k+1} = (1-\alpha)\epsilon_k. \tag{2.46}$$

On voit donc que l'erreur est réduite et dépend du choix de α. Celui-ci va contrôler la rapidité de convergence et la stabilité. En pratique, on choisira α compris entre 0.1 et 1.

Pour résumer ce que nous avons vu, voici l'algorithme alpha-LMS :

1. Initialiser le vecteur de poids $\vec{w}_{k=0}$ avec des valeurs aléatoires.
2. Répéter pour chaque nouvelle entrée en incrémentant k :
   (a) Calculer la réponse de l'adaline : $y_k = \vec{x}_k^T \vec{w}_k$,
   (b) Calculer l'erreur : $\epsilon_k = d_k - y_k$,
   (c) Calculer la mise à jour : $\triangle\vec{w}_k = \alpha\frac{\epsilon_k \vec{x}_k}{|\vec{x}_k|^2}$,
   (d) Mettre à jour les poids : $\vec{w}_{k+1} = \vec{w}_k + \triangle\vec{w}_k$.

2.5.7 L'algorithme Recursive Least Square (RLS)

Le principal inconvénient de l'algorithme est sa lenteur à converger vers une solution. L'algorithme en ligne "Recursive Least Square" [46, 12] apporte une grande amélioration, mais au désavantage d'un coût algorithmique plus important.

Contrairement au LMS, le RLS utilise la totalité des entrées passées pour calculer à chaque itération une estimation de la matrice d'autocorrélation R et celle du vecteur d'intercorrélaction $\vec{p}$. De fait, la complexité, qui était de l'ordre de $\mathcal{O}(n)$ pour l'algorithme LMS, passe à $\mathcal{O}(n^2)$ pour le RLS en raison des multiplications entre matrice et vecteur.

Pour décroître l'importance des entrées précédentes par rapport aux nouvelles, la fonction de performance est redéfinie comme suit :

$$J_k = \sum_{i=1}^{k} \rho^{k-i} \epsilon_{i,k}^2 \tag{2.47}$$

où l'erreur est calculée pour chaque entrée précédente :

$$\epsilon_{i,k} = d_i - \vec{w}_k^T \vec{x}_i \tag{2.48}$$

où le facteur ρ, appelé facteur d'oubli, est égal à 1 si l'on veut que toutes les informations précédentes soient tenues en compte de manière identique ou inférieure à 1 pour que l'importance des anciennes données décroissent de manière exponentielle.

Au temps k, la meilleure estimation de la matrice d'autocorrélation R et celle du vecteur d'intercorrélaction $\vec{p}$ est:

$$R_k = \sum_{i=1}^{k} \rho^{k-i} \vec{x}_i \vec{x}_i^T = \rho R_{k-1} + \vec{x}_k \vec{x}_k^T, \tag{2.49}$$

$$\vec{p}_k = \sum_{i=1}^{k} \rho^{k-i} \vec{x}_i d_i = \rho \vec{p}_{k-1} + \vec{x}_k d_k. \tag{2.50}$$

Comme vu précédemment à la section 2.5.3, la meilleure estimation du vecteur de poids optimal est donnée par:

$$\vec{w}_k = R_k^{-1} \vec{p}_k. \tag{2.51}$$

Soient A et B deux matrices $N \times N$ définies positives, C une matrice $N \times M$, et D une matrice $M \times M$ définie positive. Alors, il existe un lemme [46] qui nous prouve que si $A = B + CD^{-1}C^T$, alors $A^{-1} = B^{-1} - B^{-1}C(D + C^T B^{-1} C)^{-1} C^T B^{-1}$.

Nous appliquons ce lemme à l'équation 2.49 et nous obtenons:

$$R_k^{-1} = \rho^{-1} R_{k-1}^{-1} - \frac{\rho^{-2} R_{k-1}^{-1} \vec{x}_k \vec{x}_k^T R_{k-1}^{-1}}{1 + \rho^{-1} \vec{x}_k^T R_{k-1}^{-1} \vec{x}_k}, \tag{2.52}$$

$$= \rho^{-1} R_{k-1}^{-1} - \rho^{-1} \vec{k}_k \vec{x}_k^T R_{k-1}^{-1} \tag{2.53}$$

où $\vec{k}_k$ est le vecteur de gain défini par:

$$\vec{k}_k = G_k \vec{x}_k \tag{2.54}$$

et où:

$$G_k = \frac{\rho^{-1} R_{k-1}^{-1}}{1 + \rho^{-1} \vec{x}_k^T R_{k-1}^{-1} \vec{x}_k}. \tag{2.55}$$

En introduisant 2.50 et 2.53 dans 2.51 et en remarquant que $G_k = R_k^{-1}$, nous obtenons, après quelques manipulations, la règle d'adaptation:

$$\vec{w}_k = \vec{w}_{k-1} + \vec{k}_k \beta_k, \tag{2.56}$$

$$= \vec{w}_{k-1} + G_k \beta_k \vec{x}_k \tag{2.57}$$

où

$$\beta_k = d_k - \vec{w}_{k-1}^T \vec{x}_k. \tag{2.58}$$

Il faut remarquer la ressemblance avec l'algorithme LMS. Le taux d'apprentissage δ, remplacé par une matrice G_k, constitue la plus grande différence. Cette matrice peut être vue comme une matrice de contrôle optimal du taux d'apprentissage.

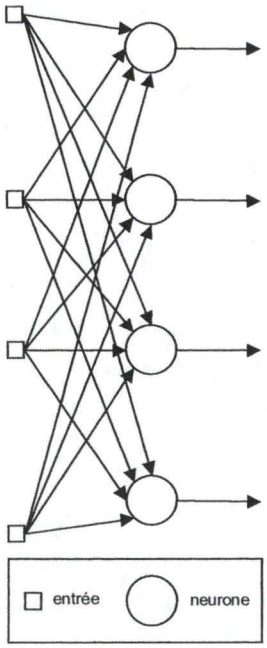


FIG. 2.8 – *Réseau simple-couche acyclique.*

2.6 Architectures des réseaux de neurones

Il existe différentes architectures de réseaux neuronaux. Celles-ci sont directement liées à l'algorithme d'apprentissage que l'on peut leur appliquer. On peut résumer ces architectures autour de trois grandes familles pour ce qui est des réseaux traditionnels.

1. **Les réseaux simple-couche acycliques**

   Ces réseaux sont constitués par une seule couche de neurones. Cette couche est composée de plusieurs neurones. Ceux-ci collectent les entrées et produisent une réponse en retour. Il n'y a pas de cycle dans cette architecture. Cette architecture est présentée à la figure 2.8.

2. **Les réseaux multi-couches acycliques**

   Dans ce type d'architecture, une ou plusieurs couches cachées sont ajoutées. Leur rôle est d'intervenir de manière utile dans la production de la réponse de la couche de sortie. Les entrées sont traitées par une première couche cachée. Les sorties de cette couche sont les entrées de la couche suivante. Celle-ci peut éventuellement être une autre couche cachée. Lorsque toutes les couches cachées ont été traversées par le flux des données, les sorties de la dernière couche sont

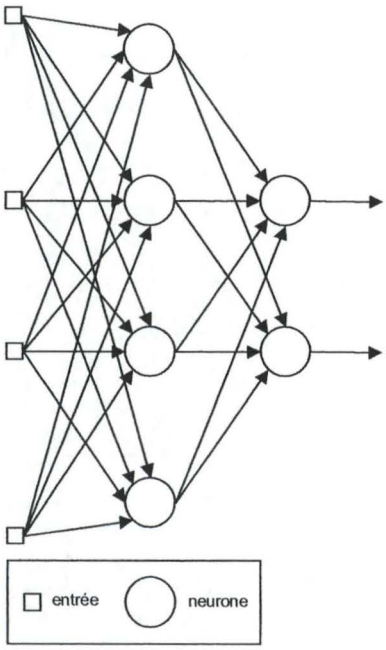


FIG. 2.9 – *Réseau multi-couches acyclique.*

traitées par la couche de sortie qui produit ainsi une réponse. Cette architecture est présentée à la figure 2.9.

3. **Les réseaux récurrents**

   Les réseaux récurrents sont sensiblement différents des réseaux multi-couches acycliques. Effectivement, ils sont constitués d'au moins une boucle de rétro-action. De plus, leur apprentissage est assuré par des algorithmes différents de ceux utilisés pour les réseaux multi-couches acycliques. On trouve, principalement, l'algorithme de rétro-propagation à travers le temps. L'aspect général d'un réseau récurrent est illustré à la figure 2.10.

2.7 Caractéristiques des réseaux de neurones

La première grande qualité d'un réseau de neurones est sa capacité à généraliser. Un réseau correctement entraîné sur un ensemble fini d'apprentissage va pouvoir étendre sa connaissance ainsi acquise à de nombreux exemples qui ne font pas partie de l'ensemble d'apprentissage.

La seconde qualité d'une majorité de réseaux de neurones réside dans leur non-linéarité. Bien que les réseaux de neurones soient plus difficile à mettre en application

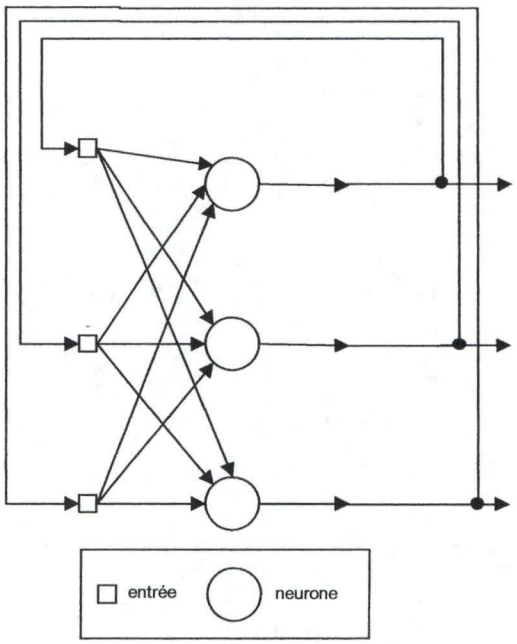


FIG. 2.10 – *Réseau récurrent. Les boucles de rétro-action utilisent un délai. Ainsi, les sorties du réseau à l'instant t sont introduites comme entrées de ce réseau à l'instant* $t + h$.

qu'une modélisation linéaire, ils procurent un énorme avantage au niveau de la modélisation de fonctions complexes ou de phénomènes très variables. Ainsi, les réseaux de neurones constituent un outil pour générer, par exemple, le signal de la parole.

La troisième qualité est la mise en correspondance des entrées-sorties. En effet, une des familles d'apprentissage consiste à fournir un apprentissage de type supervisé. Comme expliqué précédemment, on présente, selon cette méthode, un ensemble d'apprentissage constitué d'entrées et de sorties correspondantes. Ces sorties sont les réponses désirées du réseau. L'apprentissage consiste donc à réduire le plus possible l'erreur entre la sortie du réseau et la sortie désirée. Cette caractéristique ouvre, par exemple, un champ d'application particulier aux réseaux de neurones qui est la classification de "patterns".

La qualité suivante est leur capacité d'adaptation. En effet, les réseaux de neurones s'adaptent aisément à des conditions d'environnement changeantes. Cela peut être résolu en entraînant de nouveau un réseau précédemment entraîné. Dans ce cas, l'entraînement convergera plus rapidement. D'autre part, certains types de réseaux peuvent adapter leurs poids synaptiques en temps réel. Dès lors, cette qualité ouvre les champs d'application de ces réseaux à tous les domaines qui nécessitent un grand pouvoir d'adaptation tels que le traitement du signal, des phénomènes à environnement changeant, ...

La réponse motivée d'un réseau de neurones constitue la cinquième qualité. Ainsi, certains types de réseaux peuvent fournir un degré de certitude à leur réponse.

Les réseaux de neurones traitent facilement des informations contextuelles. Cela résulte de leur structure. En effet, la réponse d'un neurone influence l'action des autres.

La tolérance aux pannes est aussi un point essentiel des réseaux de neurones. Ainsi, un réseau donne toujours des résultats corrects alors que certains neurones ou liaisons neuronales sont dégradés.

Leur architecture parallèle constitue également un avantage. Cet avantage est d'autant plus important qu'il est possible d'implémenter un réseau en hardware. A ce moment, cette qualité prend toute sa dimension.

Enfin, citons, comme qualité, l'analogie biologique. En effet, la modélisation d'un neurone formel est directement inspirée de la biologie. Aujourd'hui encore, certaines recherches sont menées afin de mieux comprendre les neurones biologiques à l'aide de modélisations mathématiques.

2.8 Conclusion

Nous avons, dans ce chapitre, introduit d'une manière brève, l'environnement des réseaux de neurones. Partant de l'idée du fonctionnement du neurone biologique, nous avons vu comment nous pouvions essayer de le formaliser. Un résumé des principales architectures, ainsi qu'une explication des différentes caractéristiques des réseaux de neurones furent aussi abordées.

Enfin, nous avons développé plus longuement les éléments clefs utilisés dans la suite de ce mémoire. En effet, l'algorithme neuro-mimétique en ligne alpha-LMS sera utilisé, dès le chapitre suivant, comme technique de prédiction.

Chapitre 3

Prédiction balistique dans l'espace cartésien

3.1 Introduction

Ce chapitre possède un double objectif. Le premier but poursuivi permet de nous familiariser avec l'adaline et son apprentissage. Sur base de celui-ci, nous développerons un modèle neuro-mimétique. Ce modèle sera confronté à deux modèles physiques distincts : un modèle balistique qui fait abstraction des forces de frottement et un modèle les incluant.

Le deuxième but poursuivi consiste à déterminer le niveau d'abstraction à considérer dans notre contexte particulier d'expérimentation lié à la plate-forme robotique. Pour ce faire, nous étudierons la prédiction de la trajectoire d'un objet lancé d'une manière aléatoire dans l'espace cartésien. Nous procéderons par affinement d'un modèle physique qui décrit, de la meilleure manière possible, la trajectoire suivant un certain degré d'approximation conceptuel. Le modèle ainsi créé réalisera une approximation de la réalité destinée à s'améliorer au fur et à mesure de son évolution. Dans un premier temps, nous allons considérer un modèle uniformément accéléré où seule la gravité influencera le projectile. Nous ajouterons ensuite, à ce premier modèle, une résistance au mouvement due au frottement de l'air sur le projectile. Nous tenterons de faire apprendre ces deux modèles à des réseaux constitués d'adalines et de déterminer si nous devons tenir compte de forces telles que les forces de frottement.

3.2 Modèle de prédiction balistique simple

3.2.1 Hypothèses

Le modèle considéré au point suivant est un modèle uniformément accéléré, comme illustré à la figure 3.1. Il n'y a donc pas de force de frottement ni d'effet dû à une quelconque rotation. L'espace où évolue le mobile est l'espace cartésien. L'acquisition de données est parfaite. Nous supposons que le modèle sous-jacent peut être exprimé comme un système dynamique linéaire. Il n'y a pas de bruit de mesure sur les données. Ces hypothèses sont très importantes. En effet, elles ont pour but de simplifier, momentanément, l'approche du problème, trop complexe en leur absence.

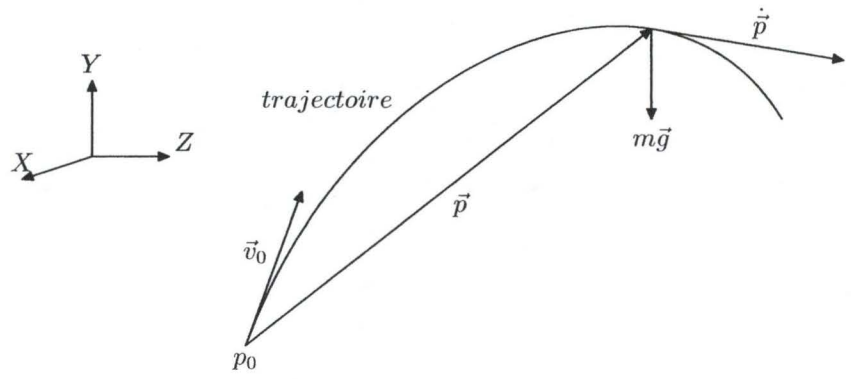


FIG. 3.1 – *Modèle uniformément accéléré d'un projectile.*

3.2.2 Modèle d'une trajectoire sans force de frottement

Le modèle de trajectoire sans force de frottement est exprimé par le jeu d'équation suivant :

$$\vec{p}(t) = \vec{p}_0 + \vec{v}_0 t + \frac{\vec{g}t^2}{2} \qquad \forall t, \tag{3.1}$$

$$\dot{\vec{p}}(t) = \vec{v}_0 + \vec{g}t \qquad \forall t, \tag{3.2}$$

$$\ddot{\vec{p}}(t) = \vec{g} \qquad \forall t, \tag{3.3}$$

avec la notation

$$\dot{\vec{p}}(t) = \frac{d\vec{p}}{dt}(t) = \vec{v}(t) \quad \text{et} \quad \ddot{\vec{p}}(t) = \frac{d\dot{\vec{p}}}{dt}(t) = \frac{d^2\vec{p}}{dt^2}(t) = \vec{g}$$

où :

$\vec{p}_0$ = la position de départ du mobile,
$\vec{p}(t)$ = la position du mobile à l'instant t,
$\vec{v}_0$ = la vitesse de départ,
$\vec{v}(t)$ = la vitesse du mobile à l'instant t,
$\vec{g}$ = l'accélération gravifique.

Le lecteur, intéressé par plus de détails sur l'élaboration de ce modèle physique, peut se référer à l'annexe A, à la page 137.

3.2.3 Principe théorique de la discrétisation

La représentation d'un système dynamique linéaire (figure 3.2) en automatique est appelée représentation d'état. Elle consiste à décrire l'état et la sortie d'un système

linéaire en fonction de son état précédent et de son entrée courante. Soient $x(t)$ l'état du système à l'instant t, $u(t)$ l'entrée du système à l'instant t et $y(t)$ la sortie de ce système à l'instant t : on peut écrire le système linéaire sous représentation d'état, en temps continu, sous la forme du système d'équations suivant:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{3.4}$$

$$y(t) = Cx(t) + Du(t), \tag{3.5}$$

avec A la matrice d'état, B la matrice d'entrée, C la matrice de sortie, D la matrice relative à $u(t)$ et $\dot{x}(t)$ la dérivée de $x(t)$ par rapport à t.

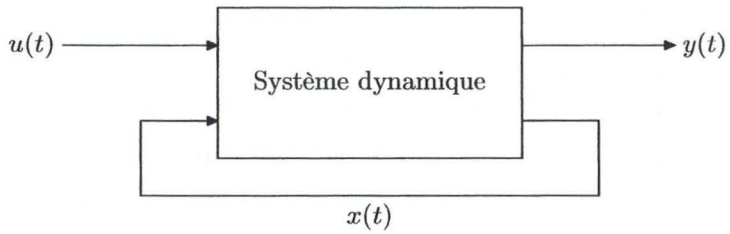


FIG. 3.2 – *Système dynamique.*

Il est intéressant de pouvoir passer d'un système linéaire en temps continu à un système linéaire en temps discret. En effet, énormément de systèmes sont observés grâce à un échantillonnage. Dans notre cas, on analyse un phénomène continu au travers de capteurs reliés à un ordinateur. Nous explorons donc un système dynamique en temps discret. Le passage en temps discret s'écrit [1] :

$$x(kh + h) = \phi x(kh) + \tau u(kh), \tag{3.6}$$

$$y(kh) = Cx(kh) + Du(kh) \tag{3.7}$$

où $k \in \mathbb{N}$, h est le taux d'échantillonnage, $\phi = e^{Ah}$ et $\tau = \int_0^h e^{As} ds B$. Pour calculer ϕ et τ, nous allons considérer que

$$\psi = \int_0^h e^{As} ds = Ih + \frac{Ah^2}{2!} + \cdots + \frac{A^i h^{i+1}}{(i+1)!} + \ldots$$

Dès lors,

$$\phi = I + A\psi,$$

$$\tau = \psi B$$

avec I la matrice identité.

Remarquons que, pour le calcul numérique de ψ, on s'arrête après le deuxième terme. Le lecteur trouvera en annexe B, à la page 141, la démonstration du passage à un système dynamique en temps discret.

3.2.4 Modèle neuro-mimétique d'apprentissage d'une trajectoire sans force de frottement

Considérons le modèle de la trajectoire, comme proposé à l'équation 3.1,

$$\vec{p}(t) = \vec{p}_0 + \vec{v}_0 t + \frac{\vec{g}t^2}{2}$$

où $\vec{g}$ représente la force pesanteur, $\vec{v}_0$ le vecteur vitesse initiale, $\vec{p}_0$ le vecteur position initiale et $\vec{p}(t)$ le vecteur position au temps t. Nous voyons que la trajectoire d'un mobile dépend de la position, de la vitesse et de la gravité. Nous voulons faire apprendre ce modèle par des adalines en identifiant le système dynamique sous une forme choisie. Nous supposons ici que nous avons un système dynamique linéaire sous-jacent. Cette hypothèse sera vérifiée de deux manières. Tout d'abord, si notre modèle neuro-mimétique donne, après apprentissage, de bons résultats en termes de performance (c'est-à-dire une faible erreur), alors, nous considérerons que cette hypothèse est vérifiée. Ensuite, pour ne laisser aucune incertitude à ce sujet, nous vérifierons théoriquement que le modèle 3.1 peut s'exprimer sous la forme du système dynamique linéaire pour lequel nous avons opté. Cette forme dépend du choix du vecteur d'état. Dans notre cas, nous sélectionnerons le vecteur d'état composé de la position à l'instant t, de la vitesse à l'instant t et d'un terme constant. Ce terme constant est utilisé pour tenir compte de la gravité. Nous supposons que nous ignorons la valeur de la gravité. Dès lors, ce sera au réseau d'adalines à l'apprendre. Nous avons donc comme vecteur d'état :

$$\vec{x}(t) = \begin{pmatrix} \vec{p}(t) \\ \vec{v}(t) \\ \vec{cst} \end{pmatrix}$$

où $t = t_0 + kh$ avec $k \in \mathbb{N}$, h le taux d'échantillonnage et t_0 l'instant initial. Le vecteur $\vec{cst}$ est un vecteur de constantes de même dimension que les vecteurs $\vec{p}(t)$ et $\vec{v}(t)$. Ce vecteur de constantes est constitué, pour la facilité de raisonnement, de constantes identiques différentes de zéro.

Nous voyons directement que nous allons avoir un problème pour exprimer la valeur de $\vec{v}(t)$ nécessaire à la constitution des ensembles d'apprentissage. En effet, cette expression de la vitesse est une mesure instantanée. Nous allons donc fournir une approximation de cette valeur. Cela aura un impact sur les coefficients des adalines. Nous mesurerons cet impact lorsque nous validerons l'apprentissage de nos adalines

de manière théorique. Choisissons donc

$$\vec{v}^A(t) = \frac{\vec{p}(t) - \vec{p}(t-h)}{h}.$$

Notre vecteur d'état vaut donc

$$\vec{x}(t) = \begin{pmatrix} \vec{p}(t) \\ \vec{v}^A(t) \\ \vec{cst} \end{pmatrix}.$$

Reprenons les équations 3.6 et 3.7 :

$$x(t+h) = \phi x(t) + \tau u(t),$$

$$y(t) = Cx(t) + Du(t).$$

Le système dynamique linéaire que nous voulons découvrir par apprentissage ne possède aucune entrée. Dès lors, le vecteur $\vec{u}(t)$, qui représente les entrées du système, est inexistant. Notre système dynamique devient donc

$$x(t+h) = \phi x(t),$$

$$y(t) = Cx(t).$$

De plus, nous voulons obtenir la position comme sortie du système. Or, cet élément est le premier composant du vecteur d'état que nous avons choisi. Nous savons donc a priori que

$$C = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}.$$

Il ne nous reste plus qu'à faire apprendre la matrice ϕ par un modèle neuromimétique. Nous sommes dans l'espace cartésien. Nous pouvons décomposer $\vec{p}(t)$ et $\vec{v}(t)$ selon leurs composantes. Le vecteur $\vec{g}$ ne pose aucun problème comme nous allons le faire apprendre par les adalines. L'équation 3.1 nous indique que nous pouvons la décomposer selon les trois coordonnées de l'espace cartésien. Nous pouvons donc considérer de manière séparée ces coordonnées. Nous obtenons trois vecteurs d'état.

$$\vec{x}_x(t) = \begin{pmatrix} p_x(t) \\ v_x^A(t) \\ cst \end{pmatrix},$$

$$\vec{x}_y(t) = \begin{pmatrix} p_y(t) \\ v_y^A(t) \\ cst \end{pmatrix},$$

$$\vec{x}_z(t) = \begin{pmatrix} p_z(t) \\ v_z^A(t) \\ cst \end{pmatrix}.$$

A partir de ces vecteurs d'état, nous pouvons construire trois systèmes dynamiques linéaires :

$$\vec{x}_x(t+h) = \phi_x \vec{x}_x(t),$$

$$\vec{x}_y(t+h) = \phi_y \vec{x}_y(t),$$

$$\vec{x}_z(t+h) = \phi_z \vec{x}_z(t).$$

Notre modèle neuro-mimétique aura pour tâche d'apprendre les matrices ϕ_x, ϕ_y et ϕ_z. Pour cela, nous allons utiliser des réseaux mono-couche d'adalines, un pour chaque matrice. L'ensemble d'apprentissage sera constitué des couples entrées-sorties à présenter à notre réseau d'adalines pour effectuer son apprentissage. En entrée, nous fournirons le vecteur d'état $\vec{x}_i(t_0+jh)$ et, en sortie, le vecteur d'état $\vec{x}_i(t_0+(j+1)h)$. Cet ensemble d'apprentissage vaut

$$\mathcal{A}_i = \{(\vec{x}_i(t_0+jh), \vec{x}_i(t_0+(j+1)h)) \mid 0 \leq j \leq T\}$$

avec $i = x$ pour le réseau d'adalines responsable de ϕ_x, $i = y$ pour le réseau d'adalines responsable de ϕ_y et $i = z$ pour le réseau d'adalines responsable de ϕ_z. $T+1$ est le nombre maximum d'exemples d'apprentissage. L'architecture du réseau responsable d'apprendre la matrice ϕ_i est représentée à la figure 3.3.

Pour réaliser l'apprentissage du réseau d'adalines i, on lui présentera successivement les couples entrées-sorties de son ensemble d'apprentissage $\mathcal{A}_i$. Cette fonction d'apprentissage est définie par

$$W_i^{new} = \mathcal{F}(\mathcal{A}_i, T, W_i)$$

où $\mathcal{F}$ est une fonction d'apprentissage définie au chapitre 2, $\mathcal{A}_i$ est un ensemble d'apprentissage, T est un taux d'apprentissage, W_i est une matrice des coefficients initiaux du réseau d'adalines et W_i^{new} est la matrice des coefficients du réseau d'adalines après

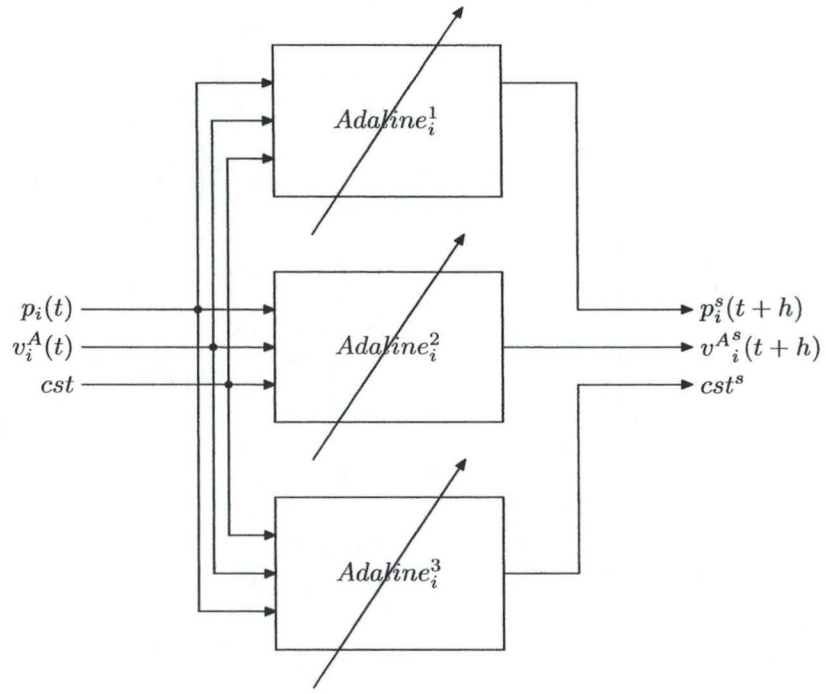


FIG. 3.3 – *Architecture des réseaux d'adalines pour l'apprentissage des matrices des systèmes dynamiques linéaires ϕ_i. Chaque réseau prend en entrée le vecteur $\vec{x}_i(t)$ et nous fournit en sortie le vecteur ligne $\vec{x}_i^s(t+h)$. L'erreur entre la sortie donnée du réseau $\vec{x}_i^s(t+h)$ et la sortie désirée $\vec{x}_i(t+h)^T$ permet d'adapter les coefficients des adalines du réseau. Les flèches en travers des adalines signifient que les coefficients des adalines sont variables en fonction de l'apprentissage.*

apprentissage. Dans notre cas, les matrices W_i et W_i^{new} sont de la forme

$$\begin{pmatrix} w_{11}^i & w_{12}^i & w_{13}^i \\ w_{21}^i & w_{22}^i & w_{23}^i \\ w_{31}^i & w_{32}^i & w_{33}^i \end{pmatrix}.$$

Les coefficients w_{j1}^i sont les coefficients du premier adaline du réseau, w_{j2}^i ceux du deuxième et w_{j3}^i ceux du troisième. Ainsi, après apprentissage, nous aurons

$$\vec{x}_i(t_0+jh)^T W_i^{new} = \vec{x}_i^s(t_0+(j+1)h) \approx \vec{x}_i(t_0+(j+1)h)^T$$

où $\vec{x}_i^s(t_0+(j+1)h)$ représente la sortie du réseau d'adalines. Nous n'avons pas l'égalité entre $\vec{x}_i^s(t_0+(j+1)h)$ et $\vec{x}_i(t_0+(j+1)h)^T$ car il reste, après apprentissage, une erreur qui est négligeable. Si l'apprentissage est suffisant et si notre système dynamique sous-

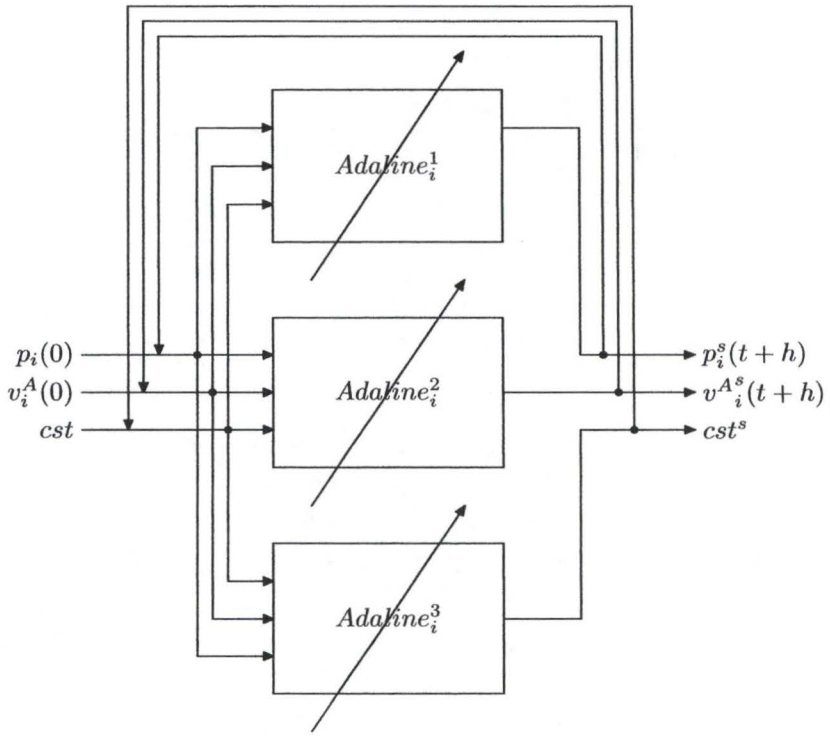


FIG. 3.4 – *Pour prédire à long terme, on donne en premier vecteur d'entrée un vecteur initial et puis, on utilise la sortie du réseau que l'on donne en entrée pour la prédiction suivante.*

jacent est bien linéaire, nous aurons

$$\phi_i \approx \begin{pmatrix} w_{11}^i & w_{21}^i & w_{31}^i \\ w_{12}^i & w_{22}^i & w_{32}^i \\ w_{13}^i & w_{23}^i & w_{33}^i \end{pmatrix} = (W_i^{new})^T.$$

3.2.5 Validation du modèle neuro-mimétique sur base de données simulées avec $Matlab_{©}$

La fonction d'apprentissage choisie pour les réseaux d'adalines est l'$\alpha - LMS$. Nos simulations sont constituées de 400 trajectoires observées pendant un intervalle de temps d'une seconde. Notre taux d'échantillonnage vaut $\frac{1}{30}$ de seconde. Nous disposons donc de 30 observations par trajectoire. Au total, nous avons 12000 observations pour constituer notre ensemble d'apprentissage. Les trajectoires sont générées aléatoirement. Pour vérifier l'apprentissage de notre modèle, nous constituons une nouvelle trajectoire aléatoire. Les résultats sont présentés aux figures 3.5, 3.6 et 3.7. Les prévisions des adalines et les erreurs sont mesurées en mètre.

Nous mesurons deux erreurs (figure 3.6). La première est l'erreur de prédiction

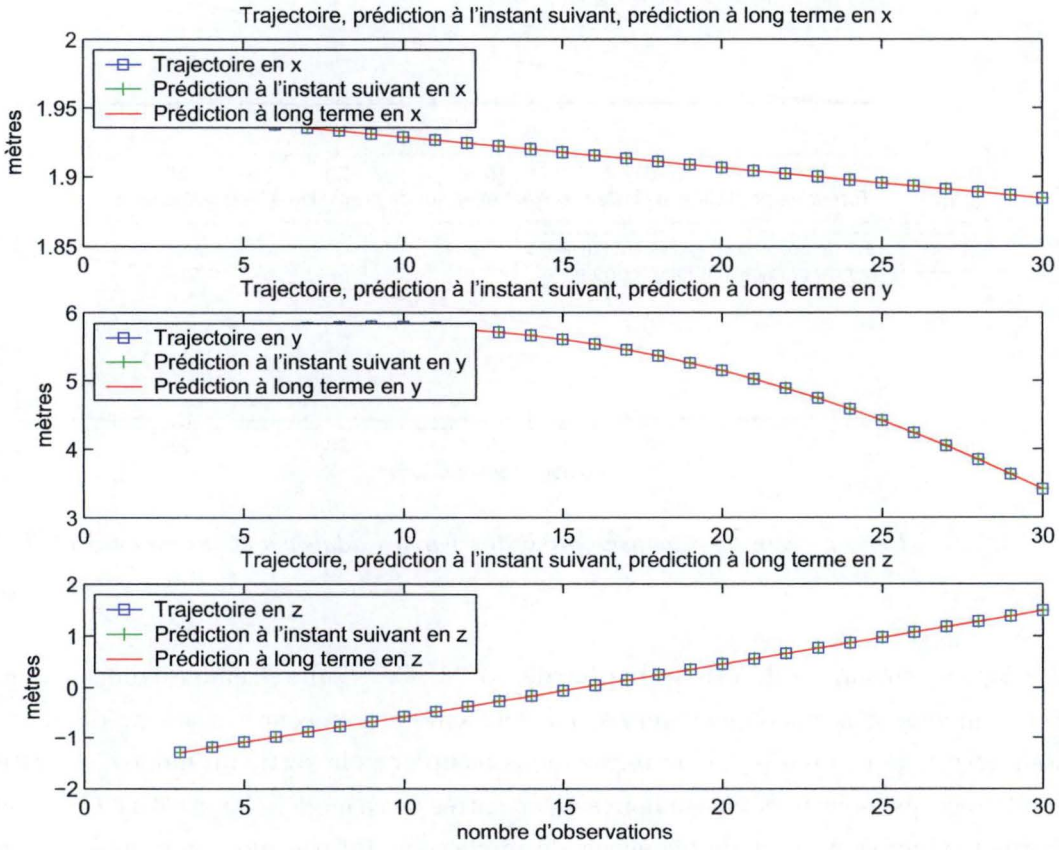


FIG. 3.5 – *Réponses des adalines sur la trajectoire de validation.*

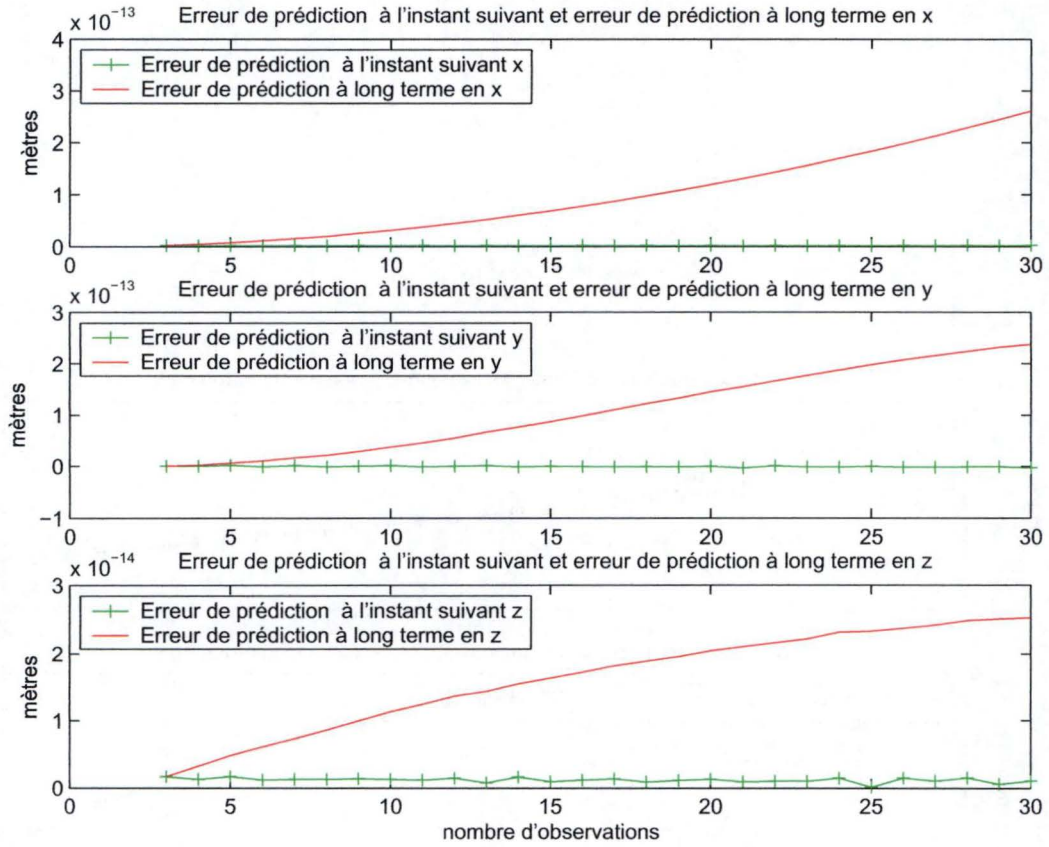


FIG. 3.6 – *Erreur entre les réponses produites par les adalines et les valeurs réelles.*

à l'instant suivant. Elle est de l'ordre de 10^{-13}. Ce résultat semble confirmer que notre modèle a correctement appris. La deuxième erreur concerne les prédictions à long terme. Pour prédire à long terme, nous récupérons la sortie du modèle constitué d'adalines que nous injectons, à nouveau, en entrée de ce modèle (figure 3.4). Cela nous permet d'incrémenter et de bénéficier de prédictions futures alors que nous n'avons pas encore les observations nécessaires pour réaliser ces prédictions futures. L'erreur, ainsi mesurée, est elle aussi de l'ordre de 10^{-13}. Cela valide le fait que notre modèle a correctement appris et qu'il est, non seulement, valable à court terme mais aussi à long terme.

Au niveau de la convergence des coefficients, les trois adalines convergent en moins de 25 trajectoires (750 exemples). Les valeurs des différentes matrices sont :

$$\phi_x^{Expérience} = \begin{pmatrix} 1.0000 & 0.0333 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix},$$

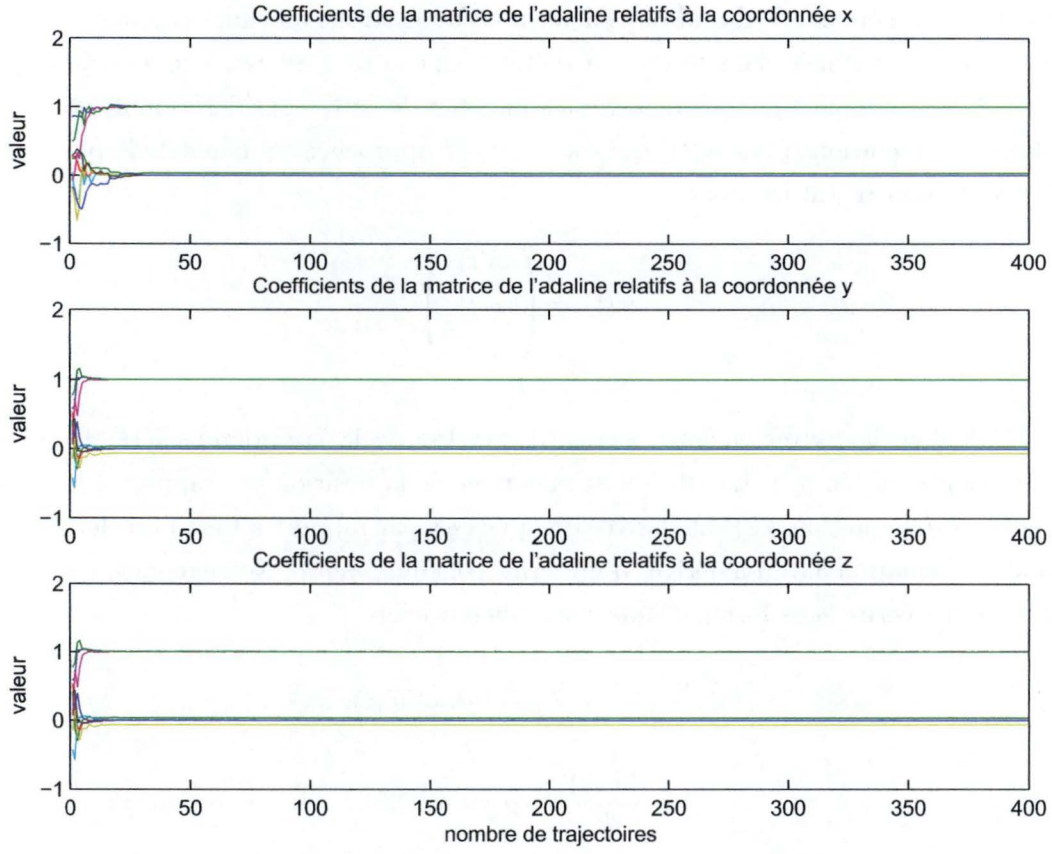


FIG. 3.7 – *Evolution des coefficients des matrices des adalines*

$$\phi_y^{Expérience} = \begin{pmatrix} 1.0000 & 0.0333 & -0.0022 \\ 0.0000 & 1.0000 & -0.0654 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix},$$

$$\phi_Z^{Expérience} = \begin{pmatrix} 1.0000 & 0.0333 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix}.$$

3.2.6 Validation théorique du modèle neuro-mimétique

Reprenons le vecteur d'état que nous avons choisi précédemment

$$\vec{x}(t) = \begin{pmatrix} \vec{p}(t) \\ \vec{v}^A(t) \\ \vec{cst} \end{pmatrix}.$$

Dans la première partie du raisonnement, nous admettrons que nous connaissons $\vec{v}(t)$, la vitesse instantanée dans le vecteur d'état même si ce n'est pas le cas en pratique. Dans la seconde partie, nous montrerons l'impact de cette considération et nous établirons les équations pour $\vec{v}^A(t)$ qui est la vitesse approchée comme définie plus haut. Notre vecteur d'état est donc

$$\vec{x}(t) = \begin{pmatrix} \vec{p}(t) \\ \vec{v}(t) \\ \vec{cst} \end{pmatrix}.$$

Calculons la dérivée de notre vecteur d'état. Pour cela, considérons $\overrightarrow{v}(t)$, la vitesse instantanée au temps t. La vitesse est la dérivée de la position par rapport à t et vaut $\overrightarrow{v}(0) + t\overrightarrow{g}$ (équation 3.2). La dérivée de la vitesse par rapport à t est l'accélération et vaut $\overrightarrow{g}$ (équation 3.3). La dérivée d'un terme constant vaut 0. Notre modèle (équation 3.1) peut s'écrire sous forme d'équations différentielles

$$\frac{d\overrightarrow{p}(t)}{dt} = \overrightarrow{v}_0 + t\overrightarrow{g} = \overrightarrow{v}(t), \tag{3.8}$$

$$\frac{d\overrightarrow{v}(t)}{dt} = \overrightarrow{g} = \overrightarrow{a}(t), \tag{3.9}$$

$$\frac{d\,\overrightarrow{cst}}{dt} = 0. \tag{3.10}$$

Nous avons supposé que notre système dynamique n'avait pas d'entrée. Nous pouvons, dès lors, écrire notre système en continu sous représentation d'état :

$$\dot{x}(t) = Ax(t), \tag{3.11}$$

$$y(t) = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} x(t), \tag{3.12}$$

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & \frac{\vec{g}}{\vec{cst}} \\ 0 & 0 & 0 \end{pmatrix}.$$

Nos simulations vont être réalisées en temps discret. En effet, les informations nous proviennent des caméras tous les $\frac{1}{30}$ de seconde ($= h$). D'après les équations 3.6 et 3.7, nous pouvons calculer

$$\psi = Ih + \frac{Ah^2}{2!} = \begin{pmatrix} h & \frac{h^2}{2} & 0 \\ 0 & h & \frac{h^2\vec{g}}{2\vec{cst}} \\ 0 & 0 & h \end{pmatrix},$$

$$\phi^{Inst} = I + A\psi = \begin{pmatrix} 1 & h & \frac{h^2\vec{g}}{2\vec{cst}} \\ 0 & 1 & \frac{h\vec{g}}{\vec{cst}} \\ 0 & 0 & 1 \end{pmatrix}.$$

Notre système d'équations, en temps discret, devient

$$\vec{x}(t+h) = \begin{pmatrix} 1 & h & \frac{h^2\vec{g}}{2\vec{cst}} \\ 0 & 1 & \frac{h\vec{g}}{\vec{cst}} \\ 0 & 0 & 1 \end{pmatrix} \vec{x}(t), \tag{3.13}$$

$$\vec{y}(t+h) = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \vec{x}(t+h) \tag{3.14}$$

où $h = \frac{1}{30}$ et $t = kh + jh$.

Nous pouvons séparer notre système selon les composantes des vecteurs et nous avons

$$\phi_x^{Inst} = \begin{pmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\phi_y^{Inst} = \begin{pmatrix} 1 & h & \frac{h^2 g}{2cst} \\ 0 & 1 & \frac{hg}{cst} \\ 0 & 0 & 1 \end{pmatrix},$$

$$\phi_z^{Inst} = \begin{pmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

car les composantes de la gravité valent 0 en x et z selon nos conventions de représentation.

Enfin, il reste un problème à résoudre. Nous avons admis que la vitesse était une vitesse instantanée, ce qui n'est pas le cas. Cela va changer les coefficients que nous venons d'établir théoriquement.

Considérons le modèle de la trajectoire, comme proposé à l'équation 3.1,

$$\vec{p}(t) = \vec{p}_0 + \vec{v}_0 t + \frac{\vec{g}t^2}{2}$$

où $\vec{g}$ représente la force pesanteur, $\vec{v}_0$ le vecteur vitesse initiale, $\vec{p}_0$ le vecteur position initiale et $\vec{p}(t)$ le vecteur position au temps t. Nous allons montrer qu'il est possible de reformuler le vecteur $\vec{v}(t)$ qui exprime la vitesse instantanée en fonction de la position et de la gravité. Nous pourrons observer la différence existant entre la vitesse instantanée et l'approximation $\vec{v}^A(t)$ que nous donnons au modèle et la

répercussion sur les coefficients.

En considérant $t = T - h$ dans l'équation 3.1, nous avons

$$\vec{p}(T-h) = \vec{p}_0 + \vec{v}_0 T - \vec{v}_0 h + \frac{\vec{g}}{2}T^2 - \vec{g}Th + \frac{\vec{g}}{2}h^2 \tag{3.15}$$

et nous savons que

$$\vec{p}(T) = \vec{p}_0 + \vec{v}_0 T + \frac{\vec{g}}{2}T^2. \tag{3.16}$$

Considérons la différence des équations 3.16 et 3.15, nous avons

$$\vec{p}(T) - \vec{p}(T-h) = \vec{v}_0 h + \vec{g}Th - \frac{\vec{g}}{2}h^2.$$

Considérons, dès lors, l'approximation de la vitesse donnée par

$$\vec{v}^A(t) = \frac{\vec{p}(T) - \vec{p}(T-h)}{h}$$

et nous obtenons

$$\vec{v}^A(t) = \frac{\vec{p}(T) - \vec{p}(T-h)}{h} = \vec{v}_0 + \vec{g}T - \frac{\vec{g}}{2}h.$$

Or $\vec{v}(T) = \vec{v}(0) + \vec{g}T$. Dès lors, les coefficients obtenus à l'aide des réseaux d'adalines vont corriger cette différence en modifiant les coefficients liés à la constante. Cette modification n'interviendra que pour la matrice ϕ_y car dans les autres cas, la valeur de différence est nulle. Nous aurons donc

$$\phi_x = \begin{pmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\phi_y = \begin{pmatrix} 1 & h & \boxed{\frac{h^2 g}{cst}} \\ 0 & 1 & \frac{hg}{cst} \\ 0 & 0 & 1 \end{pmatrix},$$

$$\phi_z = \begin{pmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Etant donné que nous avons pris $h = \frac{1}{30}\ sec$ en simulation et $cst = 5$, nous pouvons affirmer que nous avons bien appris le modèle dynamique sous-jacent (qui est linéaire). Nous avons

$$\phi_x^{Expérience} \approx \phi_x,$$

$$\phi_y^{Expérience} \approx \phi_y,$$

$$\phi_y^{Expérience} \approx \phi_y.$$

Nous ne mettons pas l'égalité car les coefficients des matrices $\phi_i^{Expérience}$ convergent vers les coefficients des matrices ϕ_i pour $i = x,y,z$.

3.3 Modèle de prédiction balistique avec frottement

3.3.1 Hypothèses

Dans ce modèle, le projectile est soumis à deux forces physiques : la gravité et le frottement de l'air sur l'objet. Nous ne considérons pas la force due à une quelconque rotation. Il n'y a donc pas de Force de Magnus. L'espace où évolue le mobile est toujours l'espace cartésien. L'acquisition de données est parfaite. Dès lors, il n'y a pas de bruit de mesure sur les données. Il n'y a que l'addition d'une force de frottement par rapport au modèle précédent. Le fait de considérer la force de frottement de l'air rend notre modèle satisfaisant, dans l'espace cartésien, pour la gamme de problèmes que nous essayons de traiter.

3.3.2 Modèle d'une trajectoire avec force de frottement

Nous modélisons la force de frottement comme étant opposée au sens de la vitesse instantanée de l'objet et possédant une norme proportionnelle au carré de cette vitesse :

$$\vec{F}_{frott} = m\vec{a}_{frott} = -\frac{1}{2}C_x\rho_0 S \|\vec{v}\| \vec{v}, \tag{3.17}$$

$$\vec{a}_{frott} = \frac{-\frac{1}{2}C_x\rho_0 S \|\vec{v}\| \vec{v}}{m} \tag{3.18}$$

où :

$$C_x = le\ coefficient\ relatif\ à\ l'objet(\simeq 0.4),$$
$$\rho_0 = la\ masse\ volumique\ de\ l'air(Kg/m^3),$$
$$S = la\ surface\ de\ l'objet\ soumise\ au\ frottement,$$
$$\vec{v} = la\ vitesse\ de\ l'objet,$$
$$m = la\ masse\ du\ projectile.$$

Pour une balle de tennis, la surface S est égale à $\frac{\pi d^2}{4}$, avec un diamètre d valant 6,70 cm. Sa masse est de 58 gr et le coefficient C_x peut être choisi, arbitrairement,

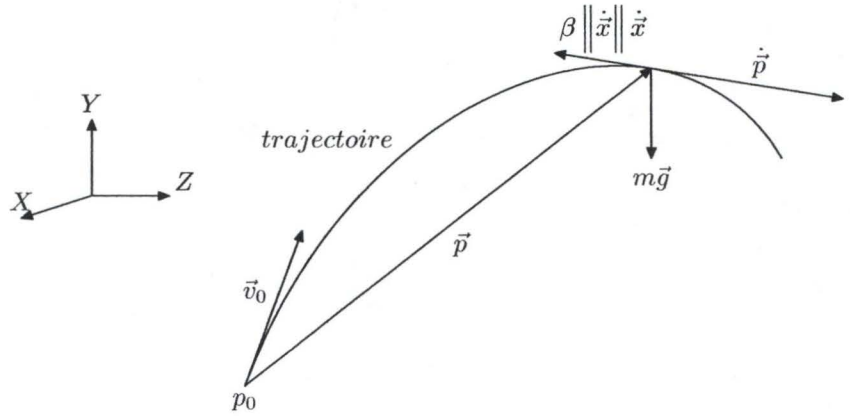


FIG. 3.8 – *Modèle d'un projectile avec force de frottement.*

égal à 0,4. Pour une pression atmosphérique et une température comparables à celles du laboratoire, nous prendrons $\rho_0 = 1.25 Kg.m^{-3}$.

La force résultante sur le mobile est devenue :

$$\begin{aligned}
\vec{F} &= \vec{F}_{grav} + \vec{F}_{frott}, \\
\vec{F} &= m\vec{a} = m\vec{a}_{grav} + m\vec{a}_{frott}, \\
\vec{F} &= m\vec{g} - \frac{1}{2} C_x \rho_0 S \left\| \vec{v} \right\| \vec{v}.
\end{aligned}$$

L'équation de la trajectoire balistique devient alors :

$$\dot{\vec{p}}(t) = \frac{d\vec{p}}{dt} = \vec{v}(t), \tag{3.19}$$

$$\ddot{\vec{p}}(t) = \frac{d\vec{v}}{dt} = \vec{g} + \beta \left\| \vec{v} \right\| \vec{v} \tag{3.20}$$

où :

$$\beta = \frac{-\frac{1}{2} C_x \rho_0 S}{m}.$$

3.3.3 Modèle neuro-mimétique d'apprentissage d'une trajectoire avec force de frottement

Considérons le problème de force de frottement différemment. Le terme β des équations 3.19 et 3.20 est de l'ordre de 0,015 m^{-1} pour une balle de tennis et est inférieur à cette valeur pour une balle de golf. De plus, la vitesse $\overrightarrow{v}(t)$ ne pourra pas être très élevée à cause du cadre d'expérimentation. Nous allons donc supposer que le terme $\beta \overrightarrow{v}(t) \| \overrightarrow{v}(t) \|$ a une influence négligeable. Nous allons faire abstraction de ce

terme, ce qui nous amène au modèle sans force de frottement décrit aux équations 3.8 et 3.9. Nos modèles sont donc

$$\vec{x}_x(t+h) = \phi_x \vec{x}_x(t),$$

$$\vec{x}_y(t+h) = \phi_y \vec{x}_y(t),$$

$$\vec{x}_z(t+h) = \phi_z \vec{x}_z(t)$$

avec

$$\phi_x = \begin{pmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\phi_y = \begin{pmatrix} 1 & h & \frac{h^2 g}{cst} \\ 0 & 1 & \frac{hg}{cst} \\ 0 & 0 & 1 \end{pmatrix},$$

$$\phi_z = \begin{pmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Nous allons donc utiliser les mêmes réseaux neuro-mimétiques que ceux qui ne considéraient pas de force de frottement pour apprendre ces matrices.

Nous allons valider ce modèle expérimentalement par simulation. Si nos adalines convergent et donnent une erreur acceptable, nous considérerons que ce modèle est correct pour la gamme de problèmes étudiés. De plus, si la force de frottement est négligeable, nous devrions obtenir les coefficients des matrices des adalines proches des coefficients théoriques établis pour le modèle sans force de frottement. Enfin, nous ferons une analyse de sensibilité du modèle à la force de frottement de l'air pour établir la limite d'application de notre modèle.

3.3.4 Résultats des simulations sur base de données simulées avec $Matlab_{©}$

Les résultats sont présentés aux figures 3.9, 3.10, 3.11 et 3.12. Nous réalisons nos simulations sur 400 trajectoires de 30 exemples. Nos trajectoires sont générées aléatoirement. Le taux d'échantillonnage vaut toujours $\frac{1}{30}$ de seconde. La fonction d'apprentissage est toujours le $\alpha - LMS$. Nous utilisons une trajectoire aléatoire pour nos mesures d'erreurs. L'erreur sur les prédictions à l'instant suivant est de l'ordre de 10^{-4} (figure 3.10), ce qui semble tout à fait raisonnable. L'erreur sur les prédictions à long terme est tout à fait acceptable pour une prédiction à long terme qui sera

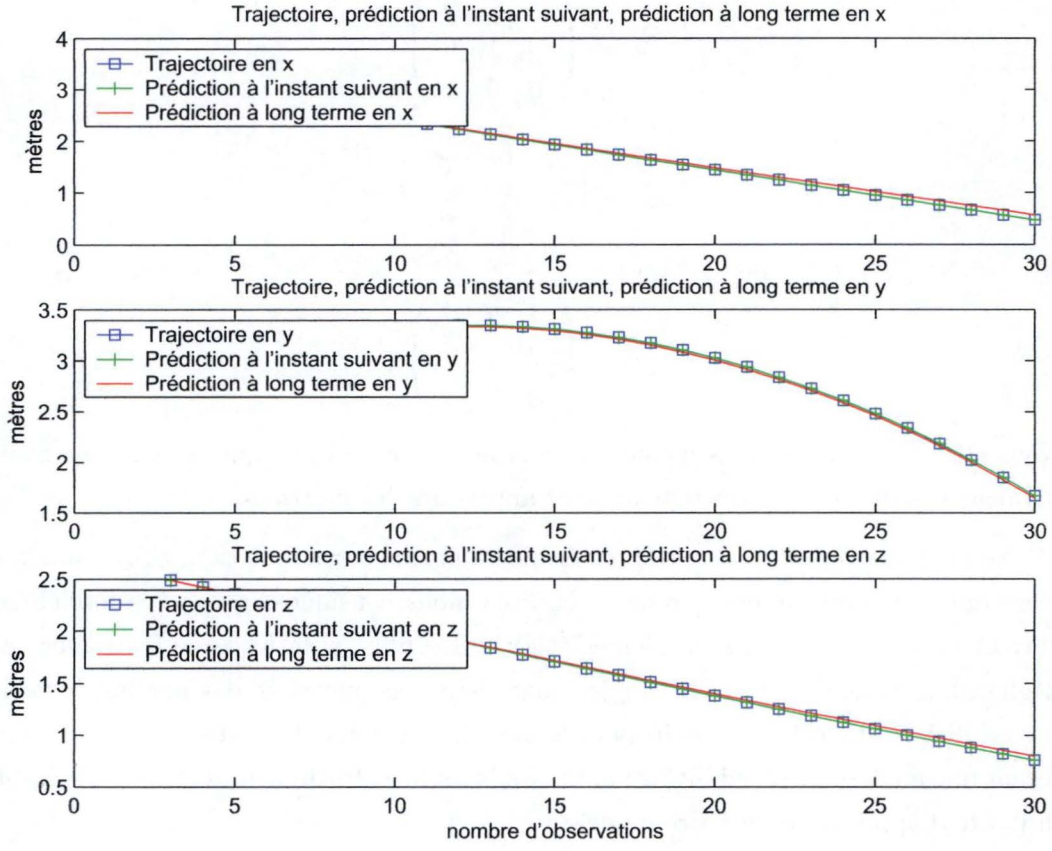


FIG. 3.9 – *Réponses des adalines sur la trajectoire de validation.*

adaptée par la suite (figure 3.11). Ce modèle semble donc approprié à la prédiction à long terme.

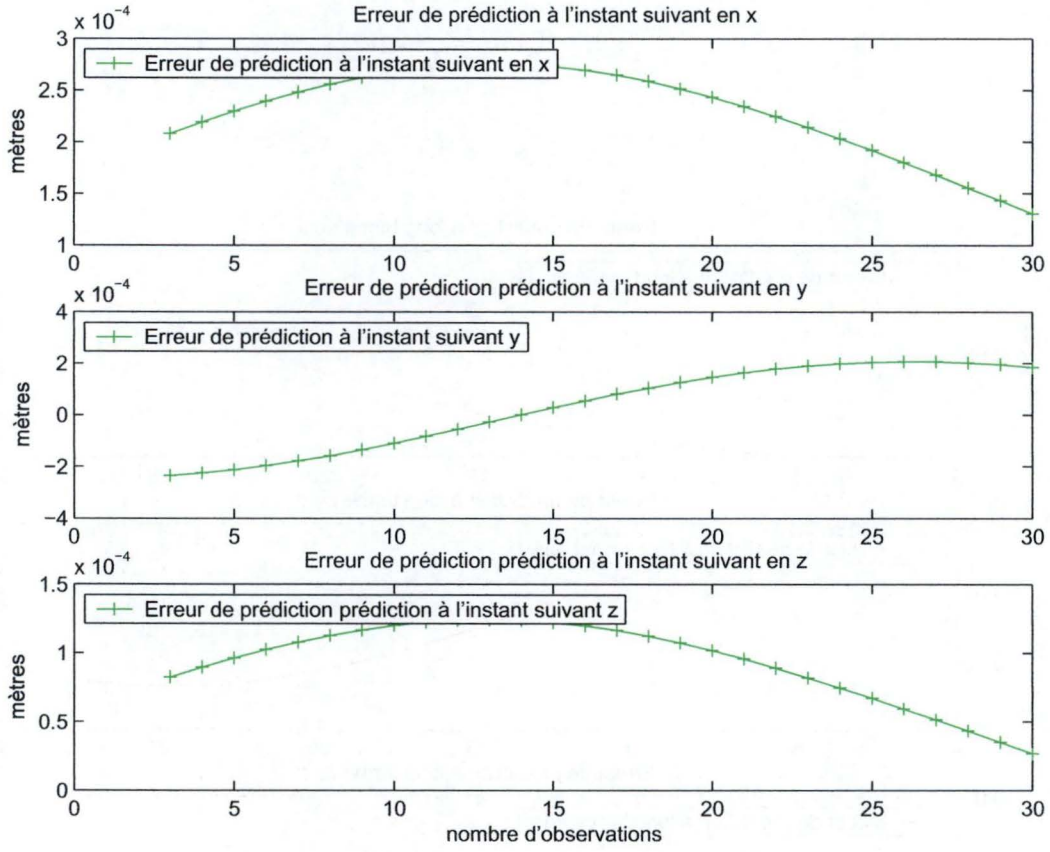


FIG. 3.10 – *Erreur entre les réponses produites par les adalines et les valeurs réelles.*

Il faut moins de 25 trajectoires pour converger (figure 3.12). Ce modèle semble tout à fait adéquat pour la prédiction avec une force de frottement faible. Comme nous l'avons supposé précédemment, nous retrouvons les coefficients théoriques.

$$\phi_x^{Expérience} = \begin{pmatrix} 1.0000 & 0.0332 & 0.0000 \\ 0.0000 & 0.9964 & 0.0002 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix} \approx \phi_x,$$

$$\phi_y^{Expérience} = \begin{pmatrix} 1.0000 & 0.0332 & -0.0022 \\ -0.0001 & 0.9961 & -0.0653 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix} \approx \phi_y,$$

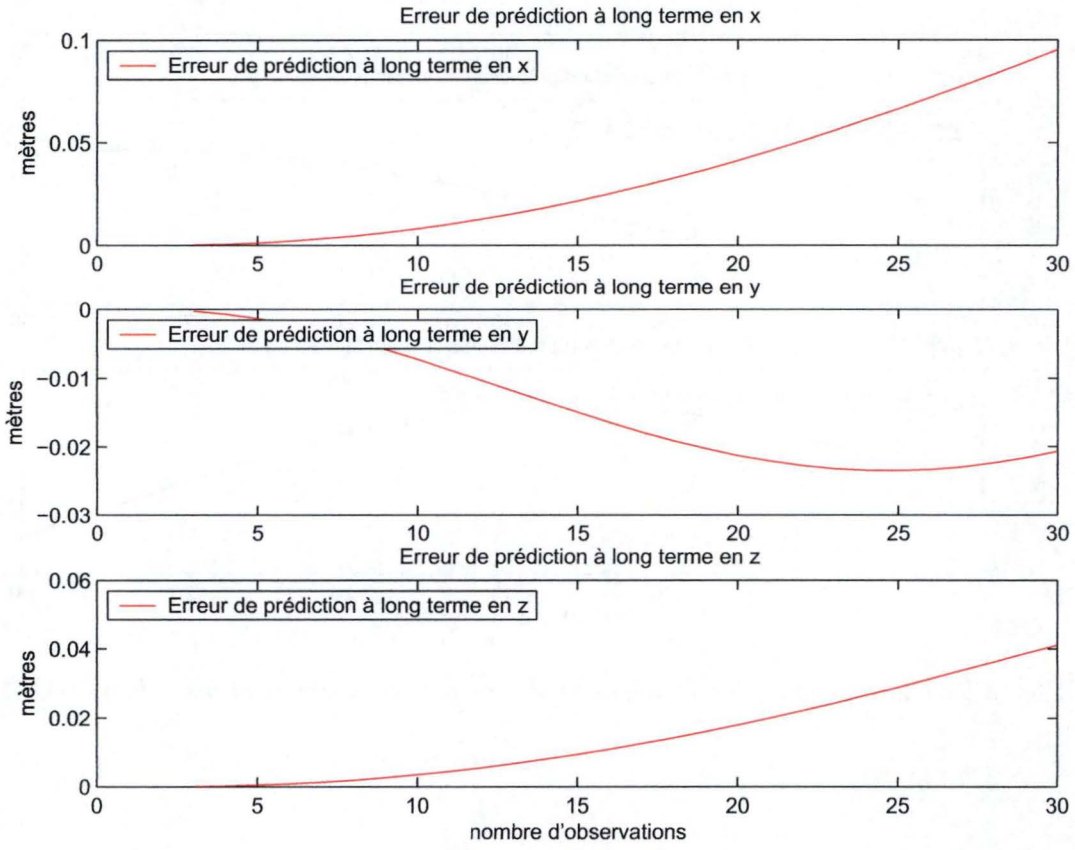


FIG. 3.11 – *Erreur entre les prédictions à long terme et les valeurs réelles.*

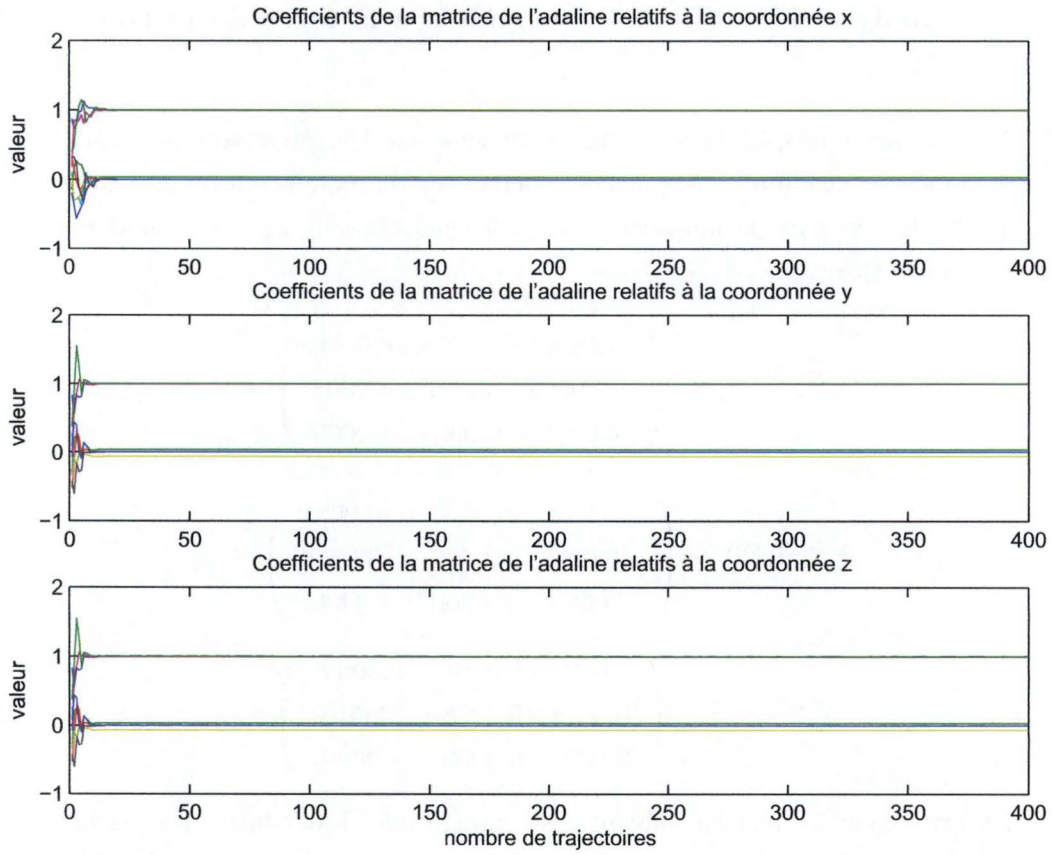


FIG. 3.12 – *Evolution des coefficients des matrices des adalines.*

$$\phi_Z^{Expérience} = \begin{pmatrix} 1.0000 & 0.0332 & 0.0000 \\ 0.0003 & 0.9954 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix} \approx \phi_z.$$

Cela semble confirmer le fait que, dans notre problème pratique, nous pouvons faire abstraction de la force de frottement.

3.3.5 Analyse de sensibilité du modèle à la force de frottement de l'air

Nous avons multiplié la force de frottement par 100 pour voir la sensibilité de notre modèle à cette force. Certains des coefficients du modèle ont du mal à converger (figure 3.13). De plus, la majorité des coefficients obtenus ne correspondent plus à notre modèle théorique.

$$\phi_x^{Expérience} = \begin{pmatrix} 1.0000 & 0.0258 & 0.0000 \\ -0.0002 & 0.7750 & 0.0003 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix} \neq \phi_x,$$

$$\phi_y^{Expérience} = \begin{pmatrix} 1.0000 & 0.0274 & -0.0031 \\ -0.0009 & 0.8225 & -0.0918 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix} \neq \phi_y,$$

$$\phi_Z^{Expérience} = \begin{pmatrix} 1.0000 & 0.0261 & 0.0000 \\ 0.0014 & 0.7818 & -0.0010 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix} \neq \phi_z.$$

La prédiction à l'instant suivant reste acceptable. Par contre, les prédictions à horizon lointain deviennent moins bonnes que précédemment. Le modèle présenté au point précédent est donc à utiliser pour des problèmes où la force de frottement est peu importante. Nous utiliserons ce modèle par la suite car nous nous trouvons dans ce cadre d'application.

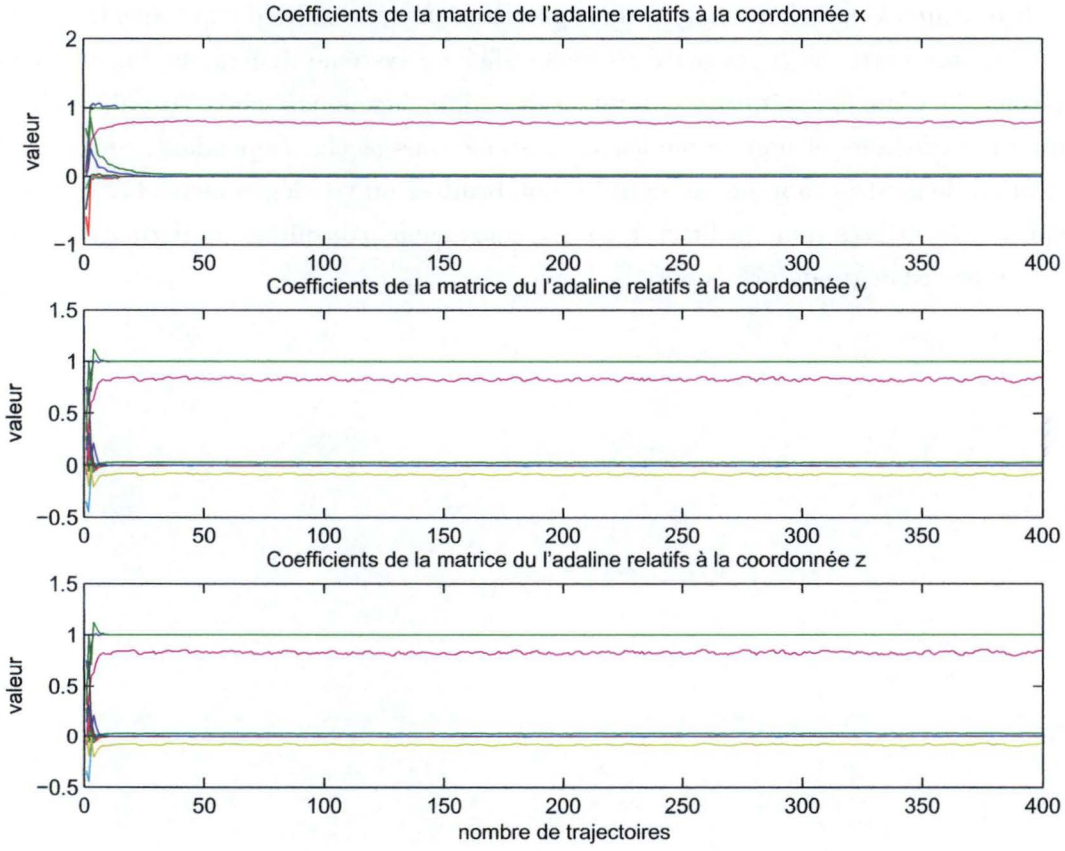


FIG. 3.13 – *Evolution des coefficients des matrices des adalines en multipliant la force de frottement par 100.*

3.4 Conclusion

Dans ce chapitre, nous avons vu qu'un modèle simplifié qui ne tient pas compte de la force de frottement de l'air, peut être adapté pour solutionner notre problème dans notre cadre d'application. En effet, étant donné que le champ d'application est la plate-forme robotique, nos lancers seront toujours assez lents et les projectiles lancés seront du type "balle de golf", "balle de tennis", ... Ce genre de projectile est communément employé dans les solutions diverses que nous avons rencontrées dans la littérature. Cependant, si nous nous plaçons dans leur cadre d'expérimentation qui est l'espace cartésien, nous arrivons à identifier un système dynamique linéaire sous-jacent alors que la littérature se limite à des approches polynômiales qui n'identifient qu'une trajectoire et non un modèle du système sous-jacent. Cependant, cette identification de modèle suppose des données non bruitées ou très légèrement. Les systèmes spécialisés utilisés dans la littérature que nous avons consultée, ne permettent sans doute pas ce degré de précision.

Chapitre 4

Vision stéréoscopique

4.1 Introduction

Notre problème de prédiction de trajectoire balistique est traité dans un cadre d'expérimentation particulier du laboratoire "TROP". Nous allons donc, dans un premier temps, décrire précisément le problème.

Ensuite, nous introduirons les concepts théoriques "clefs" qui nous permettront de développer de nouveaux modèles de prédictions. C'est pourquoi, nous présentons en détail le modèle sténopé d'une caméra.

Nous décrirons rigoureusement, dans un troisième temps, le cadre d'expérimentation du laboratoire. Cette description établira exactement la situation dans laquelle le problème se situe.

4.2 Description du problème

Considérons la prédiction de la trajectoire d'un objet lancé d'une manière aléatoire dans le confinement d'une pièce de laboratoire.

Pour prédire, il est nécessaire d'avoir un minimum d'informations pertinentes sur le chemin déjà parcouru en vue d'en extrapoler la suite. L'objet se situe dans un espace à trois dimensions. Un minimum de deux caméras est nécessaire pour déterminer, sans ambiguïté, une position à un instant donné dans cet espace. Le passage de l'espace tridimensionnel à une image bidimensionnelle entraîne, en effet, une perte d'informations. Ajouter plus de deux caméras n'apporterait pas plus d'informations pertinentes.

Nous cherchons donc à lier les informations provenant des deux caméras. Cependant, nous ne cherchons pas à retrouver la position spatiale exacte de l'objet situé dans la scène. Nous ne nous intéressons qu'à la position future de l'objet dans les images délivrées par les deux caméras ; ceci en raison de l'architecture algorithmique de la plate-forme robotique du laboratoire.

4.3 Modélisation d'une caméra perspective

Pour modéliser la vision d'une caméra le plus simplement possible, nous utiliserons le modèle sténopé, appelé aussi modèle en "trou d'épingle" ou encore modèle "pinhole" [4, 11, 27, 14]. Il s'agit de modéliser la manière dont les images passant par un point unique appelé point focal ou centre optique, se forment sur le capteur.

Le système réalise une projection perspective de la scène observée sur le plan de la caméra, le plan image π. L'équation 4.1 décrit le principe de cette projection pour tout point p situé à la coordonnée (p_x,p_y,p_z) dans le repère cartésien (X,Y,Z). Ce repère est centré au point focal comme schématisé sur la figure 4.1. La focale f est la distance entre le plan image et le centre optique.

$$p_x^a = \frac{f}{p_z}\, p_x \quad et \quad p_y^a = \frac{f}{p_z}\, p_y. \tag{4.1}$$

L'image formée du point (p_x,p_y,p_z) possède comme coordonnée (p_x^a,p_y^a) sur le plan image. Pour modéliser une caméra numérique, d'autres paramètres supplémentaires doivent être tenus en compte comme la taille des pixels du capteur et la résolution de l'image (Eq. 4.2, 4.3).

Si on ne tient pas compte de la limite de résolution, l'équation 4.1 est donc une application surjective et une fonction non-injective dans tous les cas. Autrement dit, tout point appartenant à la droite passant par le point focal et le point p est projeté sur le même point image m.

Pour comprendre la vision perspective d'une manière intuitive, imaginons un joueur de golf frappant une balle avec son club. Le joueur verra, dans un premier temps, la balle s'éloigner par rapport à lui d'une manière presque linéaire et lorsque le joueur verra la balle commencer à redescendre, celle-ci ne sera, en réalité, pas encore à son apogée. Ces deux phénomènes sont dus à la projection perspective de sa vision.

La projection perspective peut aussi être illustrée par les rails parallèles du chemin de fer dont la projection possède une intersection en un point de fuite sur le plan image [4]. Le modèle "pinhole" permet de modéliser cet aspect. Ceci est représenté schématiquement à la figure 4.2.

Modélisons maintenant une caméra numérique où des paramètres supplémentaires doivent être tenus en compte, comme la taille des pixels du capteur et la résolution de l'image. Les coordonnées en pixels (u,v) de l'objet dans l'image ont, par convention,

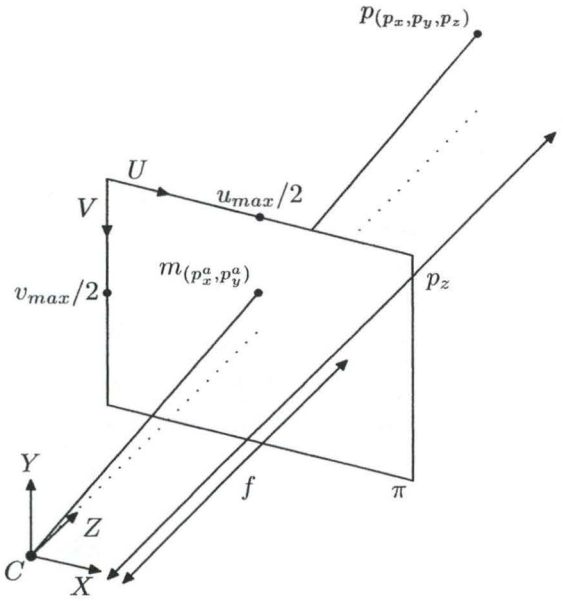


FIG. 4.1 – *Modèle sténopé. La droite en pointillé représente l'axe optique de la caméra. Le plan image est, ici, situé entre le centre optique C et la scène, mais le modèle pourrait être représenté de manière équivalente, au sens géométrique, avec le centre optique situé entre la scène et le plan image. L'image produite serait alors simplement à l'envers.*

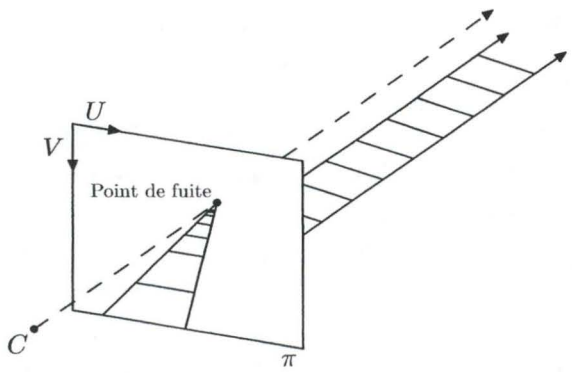


FIG. 4.2 – *Schématisation de la projection de l'intersection à l'infini des rails parallèles du chemin de fer en un point de fuite [4].*

l'origine en haut à gauche de celle-ci :

$$p_{Im_x} = u = \frac{f}{\Psi_x p_z} p_x + \frac{u_{max}}{2}, \tag{4.2}$$

$$p_{Im_y} = v = \frac{f}{\Psi_y p_z} p_y - \frac{v_{max}}{2} \tag{4.3}$$

si $\frac{f}{\Psi_x p_z} p_x \leq u_{max}$ et $\frac{f}{\Psi_y p_z} p_y \leq v_{max}$, c'est-à-dire si la résolution le permet

avec :

Ψ_x = la largeur d'un pixel en mètre,
Ψ_y = la hauteur d'un pixel en mètre,
u_{max} = la largeur en pixel de la résolution de l'image,
v_{max} = la hauteur en pixel de la résolution de l'image.

D'une manière matricielle [11, 27, 36, 14], nous pouvons décrire le modèle ainsi :

$$\begin{pmatrix} p_z u \\ p_z v \\ p_z \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{1}{\Psi_x} & 0 & \frac{u_{max}}{2} \\ 0 & \frac{1}{\Psi_y} & -\frac{v_{max}}{2} \\ 0 & 0 & 1 \end{pmatrix}}_{Para.Pixel} \underbrace{\begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{Chg.d'échelle} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} \tag{4.4}$$

où p_z intervient comme facteur multiplicateur. Pour retrouver u et v, il suffit de diviser les deux premiers coefficients de la matrice par le troisième coefficient p_z.

Cette modélisation permet de décrire en première approximation une caméra. Bien qu'elle ne modélise pas les déformations de la lentille de la caméra, nous avons jugé qu'elle est suffisamment détaillée pour notre application. Certaines approches utilisent un modèle pour des lentilles beaucoup plus complexe [27].

Il existe plusieurs moyens d'estimer la géométrie interne d'une caméra, c'est-à-dire d'estimer les éléments tels que la taille des pixels et la distance focale [4]. Le calibrage le plus simple consiste à placer un objet de géométrie connue, tel un damier, devant la caméra. Cela permet aussi de calculer une estimation de son positionnement relatif par rapport à l'objet.

Pour un système stéréoscopique où l'on veut connaître la position des objets et des caméras par rapport à un repère externe, ce qui n'est pas le cas dans nos approches, l'utilisation de la calibration est aussi très courante [36]. Il existe, cependant, d'autres techniques moins gourmandes en contraintes physiques et précisions telles que le calcul de pose, l'estimation du mouvement, la reconstruction non métrique ou encore l'auto-calibrage [4].

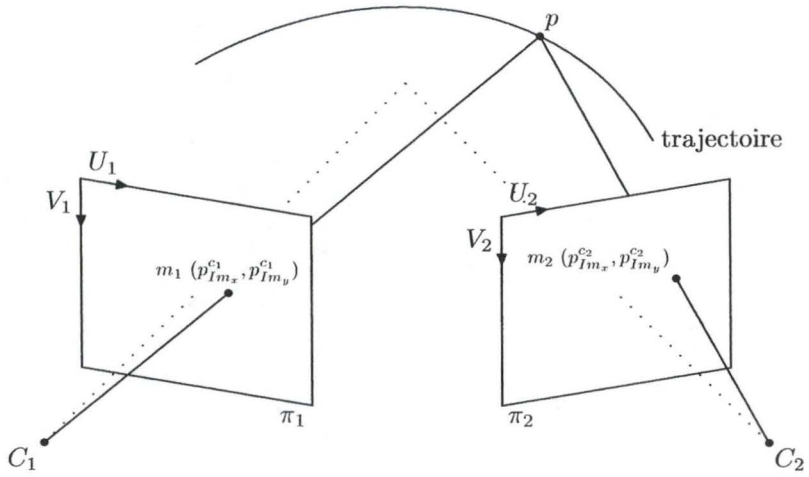


FIG. 4.3 – *Schéma illustrant la scène où l'objet p est dans le champ de vision des deux caméras. C_1 et C_2 sont les centres optiques des deux caméras. π_1 et π_2 constituent les plans images. Les droites en pointillé représentent les axes optiques des caméras.*

4.4 Description du cadre d'expérimentation

4.4.1 Cadre général

Nous disposons, pour l'expérimentation physique, de deux caméras reliées à un ordinateur permettant l'acquisition du flux d'informations. Ces caméras sont écartées d'une distance l. La première caméra forme un angle α_1 avec la base joignant les deux caméras, tandis que la deuxième forme un angle α_2 avec cette base. Le centre optique C_i sera considéré comme le centre de la caméra numéro i.

La figure 4.3 décrit, schématiquement, leur position d'une manière générale. Dans le cas où les caméras sont disposées de manière à ce que les axes optiques soient parallèles ($\alpha_1 = \alpha_2 = \frac{\pi}{2}$), la figure 4.5 permet de visionner le cône de vision commun aux deux caméras dans lequel un objet doit se situer pour être considéré par le dispositif.

Cette disposition parallèle des axes optiques permet, comme nous le verrons par la suite, d'avoir une relation simple entre la disparité d'un objet vu par les caméras et la profondeur de cet objet par rapport à la base joignant les deux caméras.

Les caméras sont disposées de manière à ce que le plan comprenant les axes optiques soit perpendiculaire au vecteur de gravité $\vec{g} = (0,g,0)^T$. Nous considérons ceci dans le but de simplifier le problème.

En effet, dans ce cas, si nous analysons l'équation d'une trajectoire dans l'espace cartésien via le modèle de la caméra (équations 4.2 et 4.3), nous pouvons observer que la gravité n'a pas d'influence dans l'équation en coordonnée X (U), mais intervient seulement en coordonnée Y (V). Cette situation peut constituer une bonne approxi-

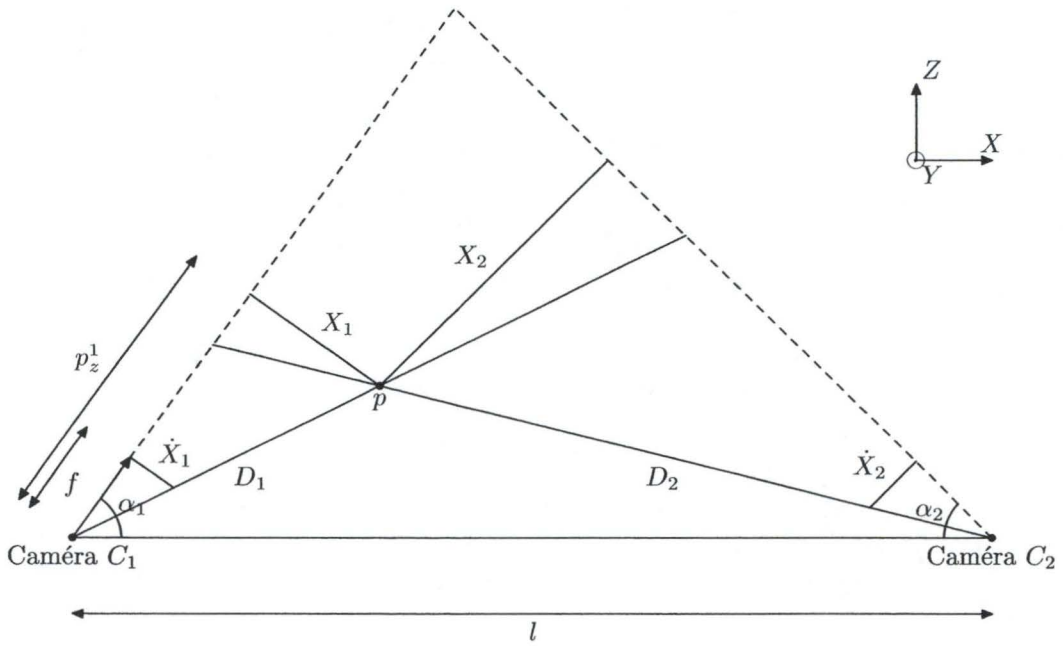


FIG. 4.4 – *Représentation de la scène, dans le plan des axes optiques, permettant le calcul de la profondeur p_z^1 du point p par rapport à la caméra numéro* 1 *dont le centre optique est en C_1. Les segments en pointillé représentent les axes optiques des caméras. Dans l'annexe C, à la page 143, nous détaillons cette figure ainsi que ces notations.*

mation tant que l'inclinaison du plan des axes optiques ne dépasse pas une dizaine de degrés.

La distance l, séparant les deux caméras, est prise assez petite en raison du cadre d'application souhaité. Ceci engendre une incertitude sur la profondeur de l'objet par rapport à la scène. Plus cette distance l sera petite, plus une faible variation des coordonnées de l'objet dans les deux images donnera une grande variation de la profondeur de cet objet.

4.4.2 Calcul de la profondeur

Le modèle sténopé, développé dans la section 4.3, est non-linéaire en raison de la division par la profondeur p_z de l'objet. Grâce à la vision stéréoscopique, le calcul de cette profondeur devient possible. Dans l'annexe C, disponible à la page 143, nous montrons le développement de ce calcul.

Pour comprendre l'équation 4.5 et son schéma 4.4, nous développons maintenant leur contexte. Considérons un objet p vu en même temps par les deux caméras. Nous souhaitons connaître la distance entre le centre optique d'une des deux caméras et cet objet. Choisissons la première, par exemple, et le plan perpendiculaire à l'axe optique de cette caméra, comprenant le point p. Avec la notation $p_{Im_x}^{c1}$ pour la caméra numéro

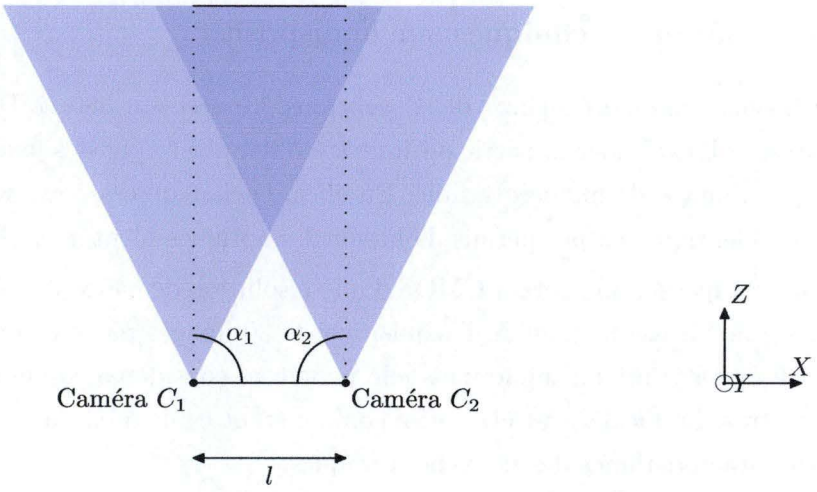


FIG. 4.5 – *Disposition des caméras vues du dessus lorsque les axes optiques sont parallèles.*

1 et $p_{Im_x}^{c2}$ pour la caméra numéro 2 (Cfr. Eq. 4.2), les coordonnées du point dans les images, nous avons :

$$p_z^1 = \frac{l(f(-\frac{p_{Im_x}^{c2}}{\Psi_x} + \frac{u_{max}}{2\Psi_x})cos(\alpha_2) - f^2 sin(\alpha_2))}{(\frac{p_{Im_x}^{c1}}{\Psi_x} - \frac{p_{Im_x}^{c2}}{\Psi_x})fcos(\alpha) + ((\frac{p_{Im_x}^{c1}}{\Psi_x} - \frac{u_{max}}{2\Psi_x})(-\frac{p_{Im_x}^{c2}}{\Psi_x} + \frac{u_{max}}{2\Psi_x}) - f^2)sin(\alpha)} \tag{4.5}$$

avec :

f	=	la distance focale identique sur les deux caméras,
l	=	la distance entre les deux centres optiques C_1 et C_2,
α_1	=	l'angle entre la base et l'axe optique de la caméra n°1,
α_2	=	l'angle entre la base et l'axe optique de la caméra n°2,
α	=	$\alpha_1 + \alpha_2$.

Dans le cas où nous plaçons les axes optiques des caméras parallèles, $\alpha_1 = \frac{\pi}{2}$ et $\alpha_2 = \frac{\pi}{2}$, nous obtenons la simplification suivante :

$$p_z^1 = \frac{lf}{\frac{p_{im_x}^{c1}}{\Psi_x} - \frac{p_{im_x}^{c2}}{\Psi_x}}. \tag{4.6}$$

Il est à noter que les équations 4.5 et 4.6 ne sont valables que dans le cas où les axes optiques des deux caméras ne sont pas gauches, c'est-à-dire qu'ils peuvent être compris dans un même plan. Cette hypothèse est peu contraignante et largement utilisée en pratique.

Nous pouvons donc, en connaissant la position d'un objet dans les deux images, les angles α_1 et α_2 des caméras et la distance l, connaître le dénominateur p_z^1 sous-jacent au modèle des caméras (Eq. 4.2 et 4.3).

4.4.3 Informations techniques sur le dispositif

Nous utilisons, comme caméras, deux webcams "Spacec@m 380 (MU2-35) USB 2.0" de marque "Trust" dont la partie optique a été remplacée par des lentilles qui ne déforment pas l'image de manière visible. En effet, l'achat de ces deux webcams de dénomination identique n'a pas permis d'obtenir des optiques identiques (Figure 4.6).

Elles sont équipées d'un capteur CMOS d'une résolution de 640 x 480 pixels (u_{max} x v_{max}) avec une vitesse maximum d'acquisition de 30 images par seconde. La taille des pixels (Ψ_x et Ψ_y) est inconnue mais elle peut être considérée comme proche de 11 x 10^{-6} mètres. La focale peut être prise comme étant égale à 12 mm si nous nous référons aux caractéristiques des nouvelles optiques.

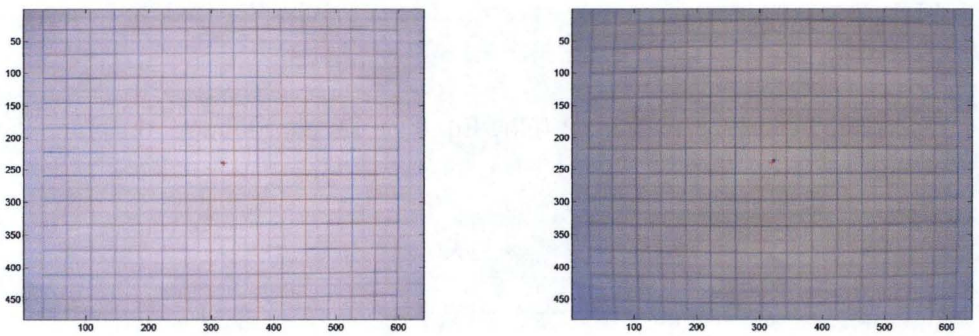


FIG. 4.6 – *A droite, une image acquise via une lentille déformant de manière visible et à gauche, une image acquise via une lentille sans déformation apparente.*

Chaque capteur est muni de son propre quartz de synchronisation. Ceci ne permet pas de synchroniser matériellement les acquisitions des caméras.

Nous avons réalisé les acquisitions sur une plate-forme Microsoft Windows© XP sp1 via l'outil VidCap de l'éditeur Microsoft (Video Capture Tool). Deux instances de cette application sont lancées en même temps pour réaliser l'acquisition.

Chaque instance est en charge d'une des deux webcams. La synchronisation se fait donc par logiciel, après acquisition, par l'algorithme de traitement d'images que nous détaillons dans le chapitre suivant.

Les architectures existantes permettant l'acquisition de flux vidéo sous Microsoft Windows© XP (Video for Windows (vfw), DirectShow et WDM Streaming) n'autorisent pas de garantir un taux d'échantillonnage constant. De ce fait, l'outil utilisé, VidCap, n'indique à la fin d'une capture que le nombre d'images non-acquises par rapport à un taux de référence paramétré à l'avance. Le taux d'acquisition varie parfois fortement sans que l'état du système ait changé.

Les paramètres influençant ce phénomène erratique de non-acquisition aléatoire

nous sont, jusqu'à présent, inconnus. Heureusement, étant donné que le logiciel donne, à la fin d'une capture, le nombre d'images réellement prises et le nombre d'images jetées pendant l'acquisition, cela nous permet de ne sélectionner que les acquisitions sans perte d'images.

4.5 Conclusion

Ce chapitre a établi le cadre de modélisation et d'expérimentation du système permettant de nous confronter à la réalité. Ainsi, nous avons développé en détail le modèle sténopé d'une caméra perspective. Nous avons donc vu que le modèle utilisé pour la modélisation des webcams est non-linéaire, ceci en raison de la profondeur de l'objet observé par rapport à la caméra. Ensuite, en décrivant en détail le cadre pratique d'expérimentation, nous avons été sensibilisés aux problèmes d'acquisition et de synchronisation des données.

Dans le chapitre suivant, nous présenterons le traitement d'images nécessaire à l'acquisition des données issues de trajectoire réelle. Nous décrirons donc les solutions apportées pour la reconnaissance de l'objet et le dispositif permettant la post-synchronisation des deux caméras.

Chapitre 5

Traitement d'images

5.1 Introduction

Dans le but de valider nos approches de prédiction sur un ensemble de trajectoires réelles, nous développons maintenant le traitement d'images appliqué aux vidéos issues du dispositif expérimental mis en place.

Nous allons premièrement développer l'algorithme de détection d'objet permettant de savoir si un objet est présent sur une image et d'en connaître précisément sa position. Ensuite, nous détaillerons le mécanisme permettant de synchroniser les images des deux vidéos après que celles-ci aient été enregistrées. Enfin, nous discuterons du cadre de réalisation dans lequel s'insère le traitement d'images.

5.2 Détection d'objet

5.2.1 Principe général

Considérons une suite d'images acquises par une caméra fixe et un objet entrant dans le champ de vision pendant l'enregistrement. Nous disposons donc d'au moins une image du décor sans la présence de l'objet; ce sera la première image de la suite. Cette image constituera l'image de référence permettant de distinguer, par comparaison, l'objet du reste de l'image.

5.2.2 Algorithme de détection

Le traitement très simple comprend plusieurs étapes successives :

1. Soustraction de l'image référentielle de l'image en traitement
2. Conversion de l'image en noir et blanc par seuillage
3. Ouverture morphologique binaire
4. Détermination du centre de l'objet trouvé

Pour la réalisation de cet algorithme, nous nous sommes aidés des fonctions existantes de "*Image Processing Toolbox*" Version 4.1 du logiciel *Matlab* © Version 6.5.1 R13 sp1.

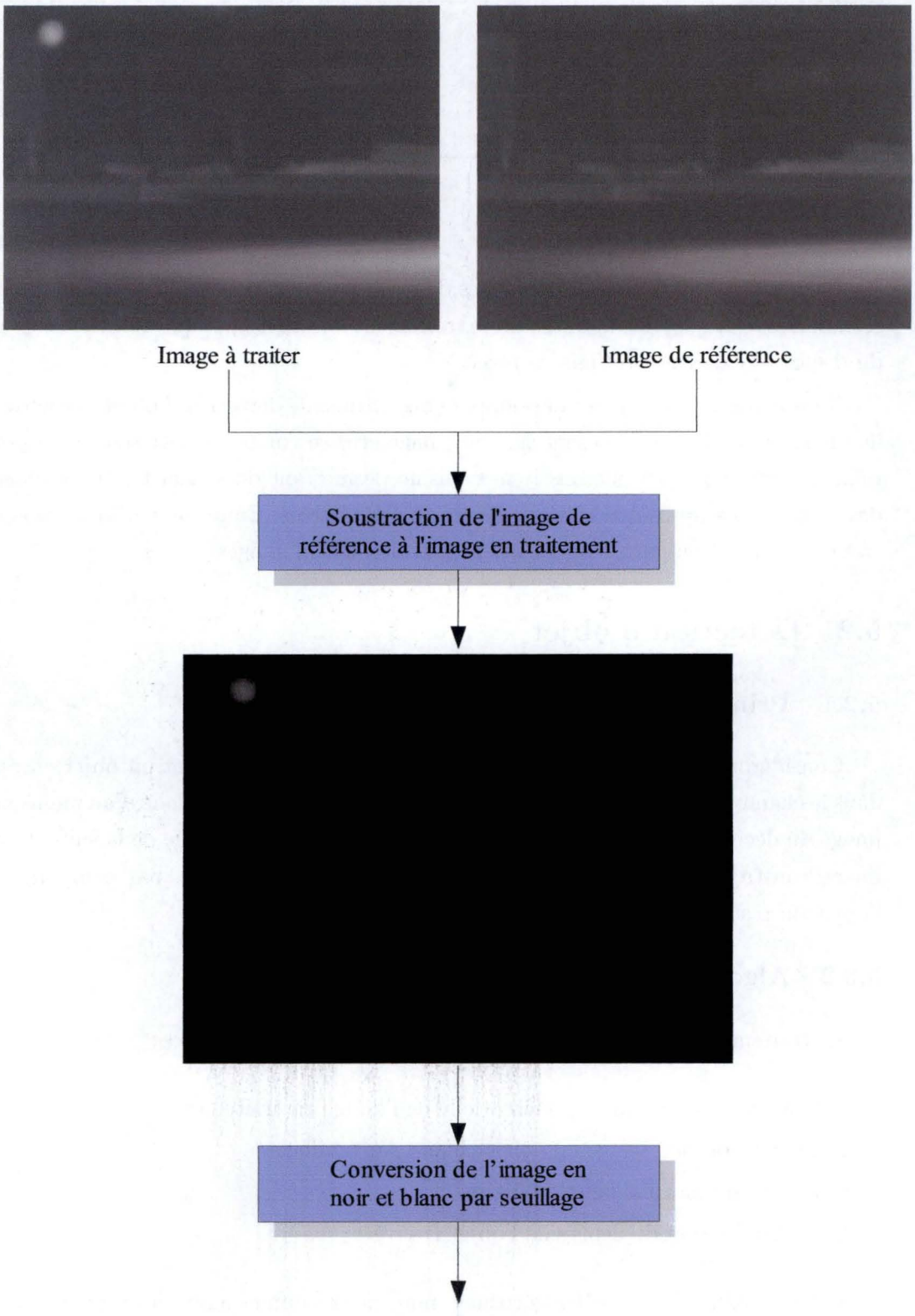
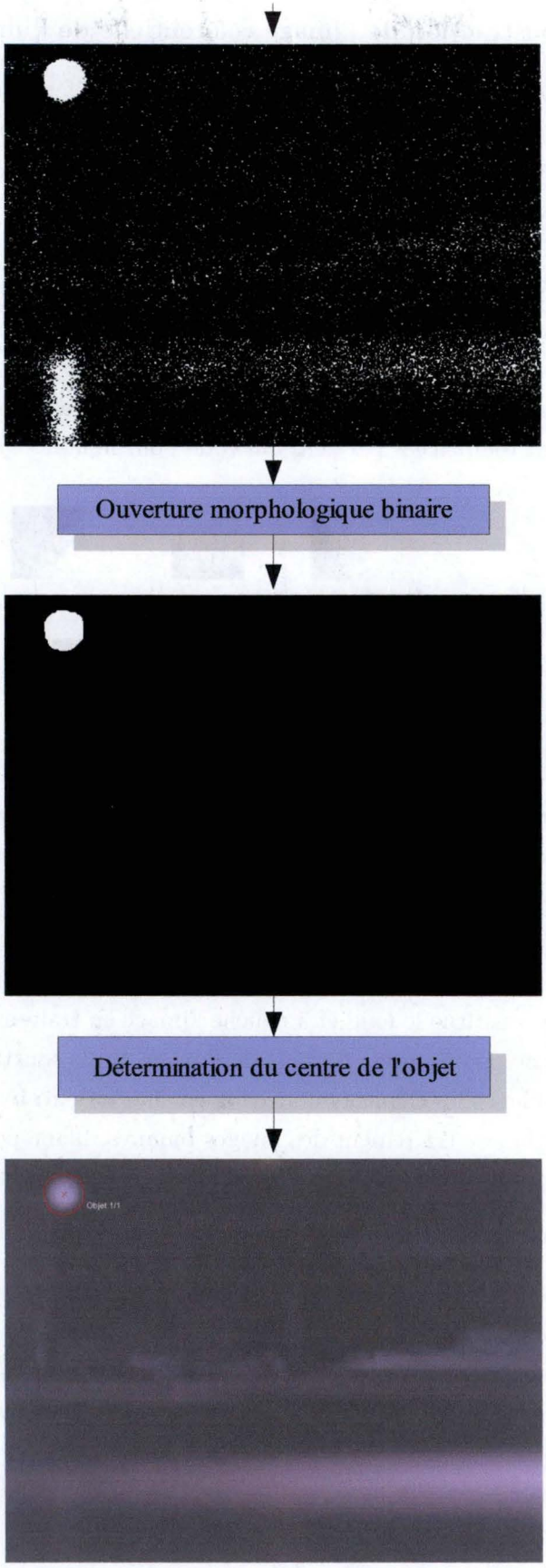
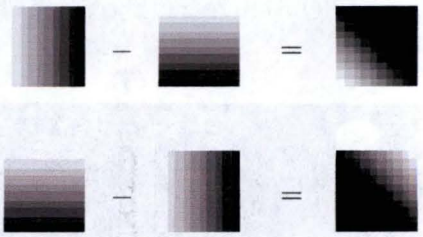


FIG. 5.1 – *Schéma de l'algorithme de détection d'objet.*


Ouverture morphologique binaire
Détermination du centre de l'objet
Objet 1/1

5.2.2.1 Soustraction de l'image référentielle de l'image en traitement

Nous commençons donc par soustraire l'image référentielle de l'image en traitement. Nous procédons dans ce sens car nous désirons obtenir les nouveaux éléments en clair sur fond sombre étant donné que la balle est blanche et le fond noir. Comme nos images sont au format RGB, chaque composante d'un pixel, c'est-à-dire rouge, vert et bleu, est encodée sur 8 bits non-signés. Un pixel blanc est défini par le code (255,255,255), un pixel noir par (0,0,0). Comme la balle est blanche, elle est représentée par des pixels proches de (255,255,255). Si nous avions réalisé la soustraction dans le sens inverse, soustraire l'image en traitement de l'image de référence, nous aurions eu une image sombre et aurions perdu la trace de la balle, car les valeurs négatives sont remplacées par zéro qui code pour le noir.



Si la balle était de couleur sombre sur fond clair, nous devrions donc soustraire l'image en traitement de l'image référentielle. Une autre manière de procéder est de réaliser le complémentaire de l'image de référence et de l'image en traitement avant la soustraction, ou encore de changer le niveau de seuillage de l'étape suivante et réaliser le complémentaire de l'image après celui-ci.

A la figure 5.1, nous avons, en haut à droite de la première page, l'image de référence qui constitue le fond et à gauche l'image en traitement où apparaît une balle de golf blanche. En dessous, il s'agit de l'image après soustraction. Cette image doit mettre en évidence les éléments nouveaux en clair et le reste en sombre ; ceci en raison de la convention de traitement des images binaires, blanc pour un objet, noir pour le fond.

5.2.2.2 Conversion de l'image en noir et blanc par seuillage

Convertissons premièrement notre image obtenue après soustraction en niveaux de gris. De cette nouvelle image, nous calculons automatiquement un seuil permettant de créer une image en noir et blanc, c'est-à-dire une image binaire. Pour ce calcul, nous utilisons la méthode d'Otsu qui minimise la variance intra-classe des pixels blancs et noirs [29]. Ce seuil dépend aussi d'une valeur entière, soustraite de l'image en traitement, permettant un réglage manuel lorsque le contraste de l'image change en raison des conditions d'expérimentation.

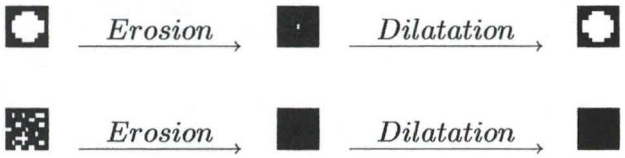
A la figure 5.1, nous avons, sur le haut de la deuxième page, l'image après seuillage. Le seuil appliqué ici n'est pas optimal, étant donné l'image après soustraction, car il ne devrait pas rendre visible autant d'éléments. Nous l'avons choisi comme tel, à titre pédagogique, pour montrer la capacité de l'algorithme. La tache en bas à gauche de l'image est la conséquence du reflet de la balle sur une surface réfléchissante.

5.2.2.3 Ouverture morphologique binaire

Nous effectuons ensuite une ouverture morphologique. Cela consiste à réaliser deux opérations successives sur l'image binaire : une érosion suivie d'une dilatation. Ces deux opérations utilisent ici, comme élément morphologique, un disque d'un rayon de 5 pixels.

Érosion morphologique binaire Pour un pixel donné, si l'un de ses voisins, c'est-à-dire un pixel compris dans l'élément morphologique centré sur le pixel en traitement, est noir, alors le pixel en traitement deviendra noir dans l'image finale.

Dilatation morphologique binaire Pour un pixel donné, si l'un de ses voisins compris dans l'élément morphologique est blanc, alors celui-ci deviendra blanc dans l'image finale.



Nous pouvons donc voir à la figure 5.1, sur le milieu de la deuxième page, l'image après ouverture morphologique. La trace du reflet de la balle, comportant des imperfections, a bien disparu. Il ne reste plus que la trace de la balle de golf.

5.2.2.4 Détermination du centre de l'objet

Nous déterminons ensuite le nombre d'éléments présents sur l'image. S'il n'y a pas d'élément présent, soit le traitement ne fonctionne pas avec le seuil donné, soit il n'y pas d'élément nouveau présent sur l'image. Si le nombre d'éléments trouvés est supérieur à un, la reconnaissance de l'objet a échoué. Sinon, il ne nous reste plus, alors, qu'à déterminer le centre de l'élément ainsi trouvé.

Enfin, à la figure 5.1, sur le bas de la deuxième page, nous avons l'image originale avec, en superposition, la position estimée du centre de gravité de la balle. La balle est de masse homogène. De ce fait, le centre de l'élément correspond bien au centre de gravité de la balle, ce qui finalement nous intéresse.

5.3 Post-synchronisation des caméras

En raison de l'indépendance physique et de l'acquisition individuelle des caméras de notre dispositif expérimental, nous sommes contraints de post-synchroniser les deux vidéos acquises, c'est-à-dire de retrouver la correspondance temporelle entre les images acquises par les deux caméras.

Étant donné que les deux caméras sont identiques physiquement et que nous appliquons les mêmes paramètres de configuration à celles-ci, nous pouvons raisonnablement considérer que le temps de constitution d'une image par une caméra est identique au temps de constitution d'une image par l'autre caméra et de plus que ce temps est constant.

Pour retrouver de manière automatique la correspondance temporelle entre les deux vidéos, nous utilisons une petite lampe placée dans la scène observée par les deux caméras. Cette lampe est visible en bas à droite à la figure 5.4. Juste avant l'arrivée de la balle dans le champ de vision d'une des deux caméras, nous allumons brièvement cette lampe. Nous possédons, comme cela, une référence temporelle pour synchroniser les deux suites d'images. Cependant, cette méthode de synchronisation ne permet de garantir qu'une intersection temporelle entre deux images et non, d'observer exactement au même instant la balle dans les deux images. Il y aura donc, au pire, une différence de temps identique au temps d'échantillonnage. Cet élément sera abordé en détails au chapitre 8.

A la fin du passage de la balle, nous réalisons une seconde reconnaissance de l'activation de la lampe, mais cette seconde référence n'est utilisée qu'à titre de validation. Pour détecter l'activation de la lampe, nous utilisons l'algorithme nous permettant de trouver la balle. Il n'y a que la valeur entière du seuil qui est différente. En réalité, nous détectons l'apparition du flash de la lampe sur la suite d'images, car le flash, plus lent que le temps d'échantillonnage, est présent sur plusieurs images successives.

Nous détaillons maintenant le schéma 5.2. Sur ce schéma, chaque élément de décision, représenté par un hexagone aplati, utilise l'algorithme de détection d'objet pour décider de la suite des opérations. Nous commençons donc par parcourir les deux suites d'images à la recherche de l'apparition du flash. A partir du moment où la référence temporelle est trouvée de chaque côté, nous incrémentons identiquement les images des deux vidéos. Après avoir recherché la disparition du flash sur les deux suites, nous attendons la présence de la balle dans le champ de vision commun aux caméras. Une fois l'objet présent sur les deux images et sans que celui-ci ne soit en contact avec les bords d'une image, nous extrayons les coordonnées de l'estimation de son centre. Après sa disparition sur l'une des deux images, nous cherchons le second flash de la lampe pour valider la correspondance des images. Si celui-ci apparaît sur une image

avant l'autre, la synchronisation n'est pas correcte.

5.4 Réalisation

Dans le but d'obtenir une meilleure synchronisation que précédemment, c'est-à-dire au mieux une désynchronisation au pire égale au temps d'échantillonnage (1/30 de seconde comme décrit au chapitre 4), nous avons utilisé une lampe stroboscopique réglée à la même fréquence que la fréquence d'acquisition des caméras. Le principe est d'éclairer la pièce plus brièvement que le temps d'intégration d'une image par les capteurs. De cette manière, le capteur de chaque caméra ne percevra la balle que pendant le flash de la lampe. Les caméras enregistrent donc l'information utile au même moment. La synchronisation est alors parfaite.

Cependant, la réalisation de ceci est plus difficile qu'il n'y paraît. En effet, il faut que la fréquence des deux caméras et celle de la lampe soient parfaitement identiques ou, en tout cas, quasi identiques pendant un laps de temps assez important. Or, dans la pratique, ce n'est malheureusement pas le cas. De plus, les caméras ne commencent pas leur acquisition d'images en même temps et le temps de constitution d'une image n'est pas réellement identique sur les deux caméras.

En raison de ces problèmes, nous n'avons pas pu utiliser à grande échelle ce moyen de synchronisation. Seules quelques trajectoires ont pu être traitées par ce moyen qui nous semble prometteur dans le cas d'améliorations futures.

En définitive, nous avons donc réalisé les captures utilisées pour la validation de nos modèles sans le stroboscope et à la lumière du jour . De fait, comme déjà expliqué, nous ne pouvons garantir qu'une intersection temporelle entre deux images. Cela conduit à une désynchronisation non-négligeable des acquisitions entre les caméras.

De plus, sans une utilisation du stroboscope, la durée d'intégration d'une image par un capteur CMOS est importante et la balle en mouvement apparaît, alors, déformée sur chaque image. Cette déformation est très bien visible à la figure 5.4. Ce phénomène est pourtant sans importance dans notre cadre d'utilisation actuel, car, pour calculer le centre de gravité, nous prendrons toujours le centre de l'élément. Étant donné que le temps d'intégration est supposé fixe, nous ne commettrons pas d'erreur en considérant la balle comme étant située au milieu de la traînée.

Il est à remarquer que cette déformation est intéressante d'un certain point de vue, car elle pourrait être mise à profit pour augmenter artificiellement le taux d'échantillonnage ou encore pour donner une indication de vitesse de l'objet en déplacement. Cependant, par manque de temps, nous n'avons pas pu développer ces approches et nous laissons ces optimisations, encore une fois, pour des améliorations futures.

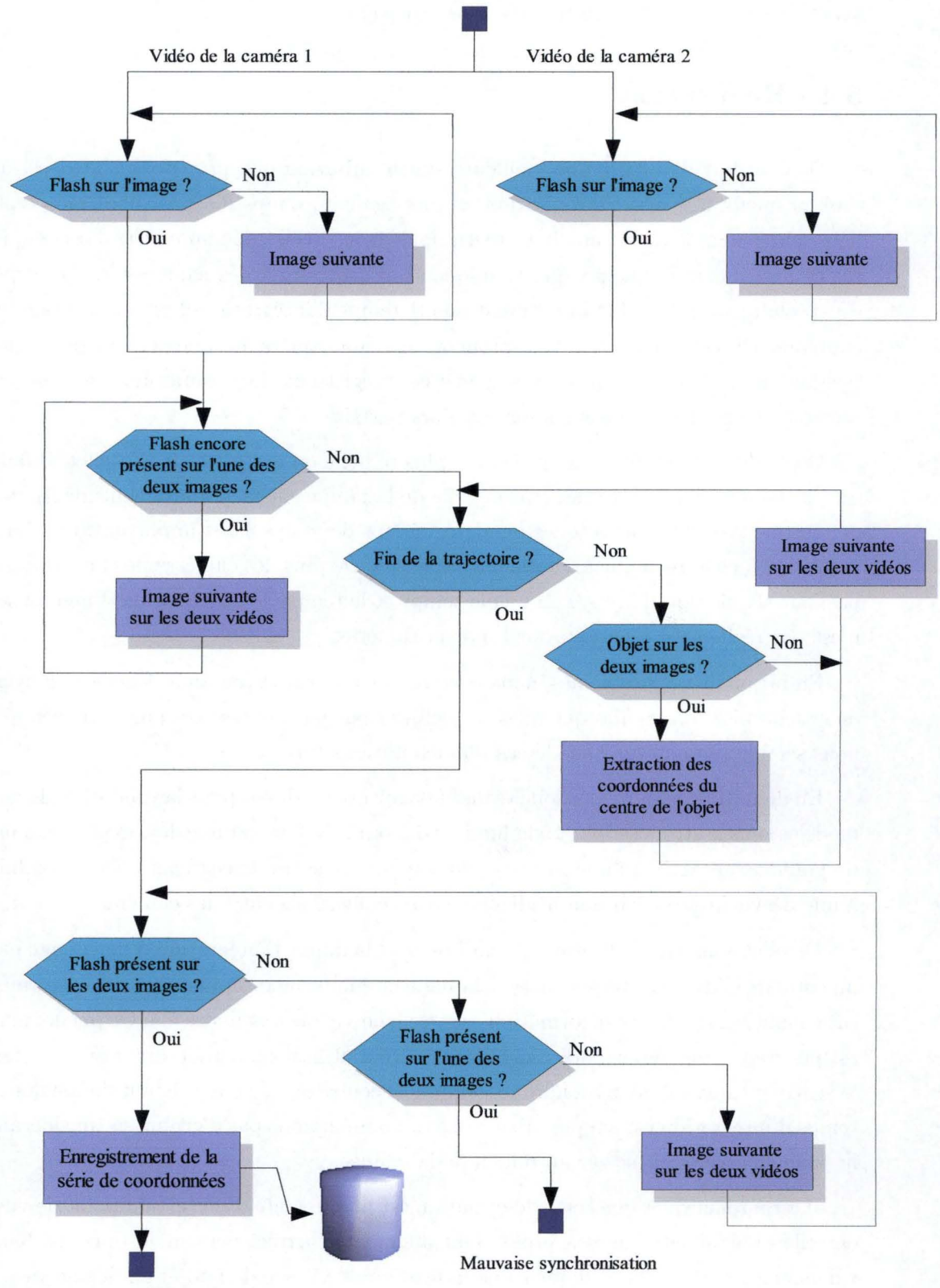


FIG. 5.2 – *Schéma de l'algorithme de post-synchronisation. Chaque élément de décision utilise l'algorithme de détection d'objet.*

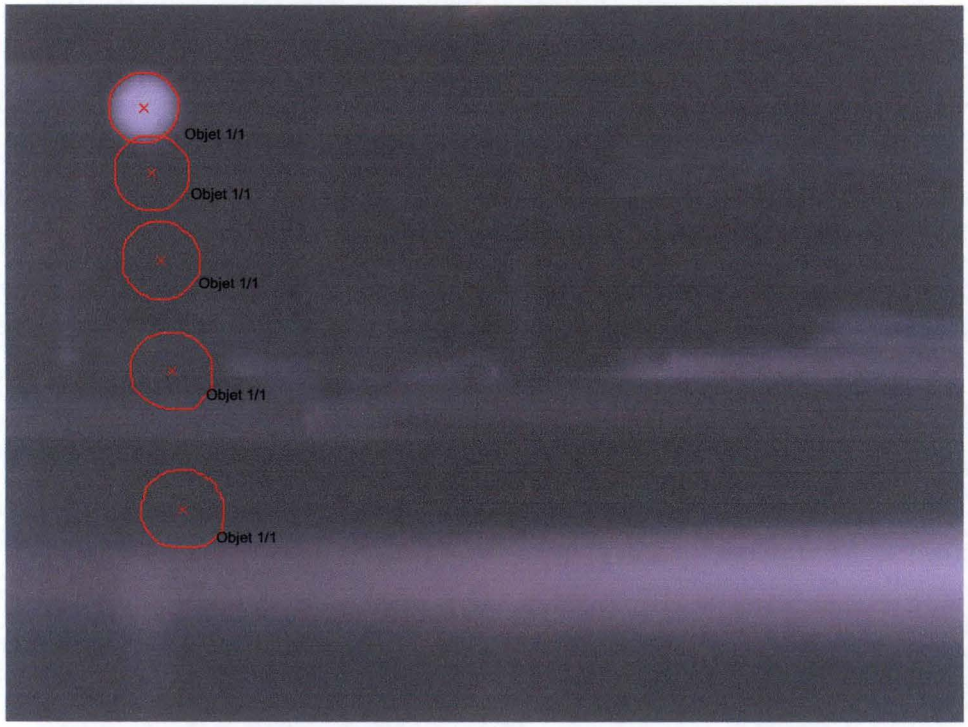


FIG. 5.3 – *Suite d'échantillons réalisée avec le stroboscope. La position estimée du centre de gravité de la balle de golf est marquée d'une croix rouge.*

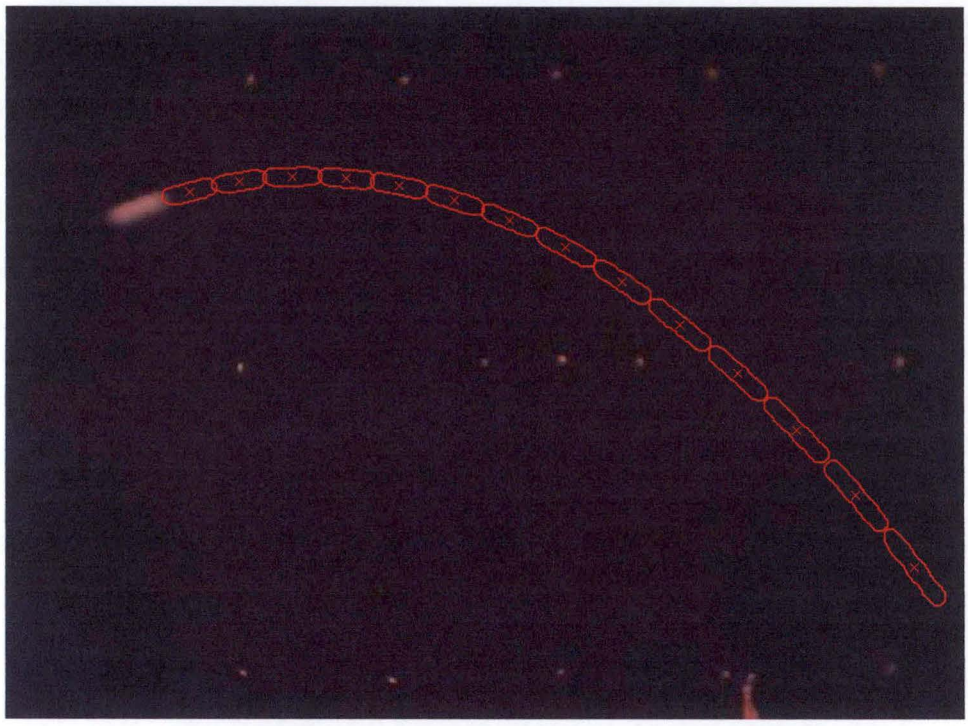


FIG. 5.4 – *Suite d'échantillons réalisée sans le stroboscope. Nous pouvons observer une déformation de la balle de golf en mouvement.*

5.5 Conclusion

Nous avons donc détaillé en profondeur la manière utilisée pour réaliser plusieurs échantillons de trajectoires balistiques à l'aide du dispositif expérimental mis en place au laboratoire. Ces échantillons seront utilisés dans les deux chapitres suivants pour tester et valider les différentes approches de prédiction imaginées.

Le traitement d'images que nous avons décrit dans ce chapitre est relativement simple et sujet à beaucoup d'améliorations futures. Ces améliorations possibles auraient pour but, soit d'améliorer sa robustesse avec, par exemple, une meilleure résistance au faible contraste entre l'image de référence et la balle, soit d'améliorer sa rapidité de traitement avec, par exemple, l'utilisation de régions d'intérêt (ROI), ou soit encore, d'extraire plus d'informations de la suite d'images avec, par exemple, l'utilisation des traînées.

Enfin, comme dans le chapitre précédent, nous avons été sensibilisés aux divers problèmes induits par les méthodes et dispositifs utilisés. Ces limites seront approfondies dans le chapitre 8.

Chapitre 6

Modèle linéaire de prédiction balistique - Caméras parallèles

6.1 Introduction

Suite à notre approche développée dans le chapitre 3 et, après avoir présenté la modélisation d'une caméra, le cadre d'expérimentation et le traitement d'images, nous continuons de complexifier les expériences en utilisant, maintenant, une modélisation plus complète de la réalité expérimentale. Ceci permet de nous confronter à la prédiction de trajectoires réelles (non simulées) acquises par nos deux caméras.

Ce chapitre a pour premier objectif de savoir si nous pouvons faire apprendre le modèle de trajectoire dans les images. A défaut de pouvoir faire apprendre complètement le modèle de trajectoire dans les images, nous tenterons de faire apprendre les éléments relatifs à la scène qui sont supposés inconnus (par exemple : l'écartement entre les caméras, la distance focale, ...). Ainsi, nous respecterons toujours les objectifs du laboratoire, c'est-à-dire faire apprendre la scène par l'expérience. De plus, nous répondrons en partie à l'objectif d'identification de système. En effet, si nous arrivons à faire apprendre, à notre modèle neuro-mimétique, les caractéristiques de la scène qui interviennent dans ce modèle alors nous aurons identifié une partie de celui-ci.

Le second objectif de ce chapitre sera de confronter notre modèle à des données réelles pour vérifier si notre modèle est adapté à la prédiction à court terme et à long terme.

6.2 Hypothèses

La disposition parallèle des axes optiques des caméras (figure 6.1) constitue la première hypothèse pour le développement du modèle de ce chapitre. Ainsi, la disparité exprime la mesure de profondeur par rapport au plan contenant les deux "plans-images" des caméras. Cette première hypothèse simplifie considérablement le raisonnement. En effet, la fonction de profondeur, décrite au chapitre 4, dans l'équation 4.5, devient :

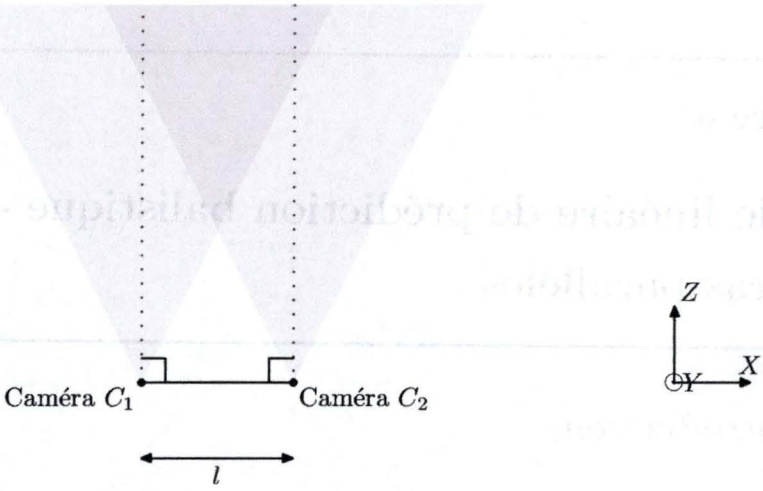


FIG. 6.1 – *Vue du dessus de la scène.*

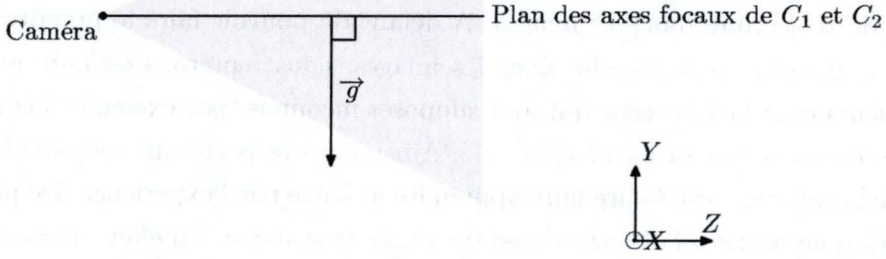


FIG. 6.2 – *Vue latérale de la scène.*

$$p_z = \frac{lf}{\frac{p_{im_x}^{c_1}}{\Psi_x} - \frac{p_{im_x}^{c_2}}{\Psi_x}}.$$

La seconde hypothèse est que le plan, contenant les axes optiques, est perpendiculaire au vecteur gravité (figure 6.2). Cette hypothèse permet d'annuler l'impact de la gravité sur la coordonnée en x dans l'image. Cela permet aussi de simplifier les raisonnements. En effet, l'équation de trajectoire en p_x devient :

$$p_x(t) = p_x(0) + v_x(0)t.$$

Enfin, la troisième hypothèse est constituée par le fait que nous considérons tou-

jours la force de frottement comme ayant une influence faible sur la trajectoire. Le chapitre 3 explique ce point et montre que cette hypothèse est raisonnable.

Ici, nous considérons le modèle "pin-hole" simplifié, c'est-à-dire celui qui ne fait pas intervenir la taille des pixels (ψ_x et ψ_y). Ceux-ci sont des facteurs d'échelle et $umax$ et $vmax$ sont de simples facteurs de translation. Ceci simplifie les raisonnements et clarifie les développements mathématiques.

6.3 Modèle neuro-mimétique d'apprentissage balistique

Nous avons vu au chapitre 4 que le modèle développé au chapitre 3 allait devenir fortement non-linéaire. En effet, ces chapitres ont montré que le modèle perspective entraîne une non-linéarité en raison de la profondeur. Or, l'adaline ne permet de traiter que des modèles linéaires. Nous avons exploré, lors de notre stage au laboratoire "Trop", plusieurs voies, notamment, en "découpant" l'espace en tranches et en changeant de modèle pour chaque tranche ("switching de modèle"). Le but était de réaliser une approximation du modèle non-linéaire avec un ensemble de modèles linéaires. Malheureusement, ces approches se sont avérées infructueuses.

Nous sommes revenus sur l'idée d'un seul adaline. Nous avons supposé que nous avions un modèle de la forme

$$\vec{x}(t) = \phi(t-h)\vec{x}(t-h) \tag{6.1}$$

où $\phi(t-h)$ est une matrice qui change au cours du temps. Pour plus de facilité, appelons cette matrice: une matrice "variable". Si les facteurs non-linéaires qui rendent la matrice du modèle "variable" pouvaient être connus et calculés, il serait peut-être possible de les sortir de cette "matrice variable" afin d'obtenir, de nouveau, un modèle linéaire. Il serait de la forme

$$\vec{x}_{OUT}(t) = \phi_i \vec{x}_{IN}(t-h) \tag{6.2}$$

avec $i = x,y$ les coordonnées de l'image d'une caméra.

Posons,

$$\vec{p}_{Im}(t) \stackrel{Def}{=} \frac{f}{p_z(t)}\vec{p}(t), \tag{6.3}$$

$$\vec{v}_{Im}(t) \stackrel{Def}{=} \frac{\vec{p}_{Im}(t) - \vec{p}_{Im}(t-h)}{h}. \tag{6.4}$$

Nous aimerions un vecteur d'état qui vaudrait

$$\vec{x}(t) = \begin{pmatrix} \vec{p}_{Im}(t) \\ \vec{v}_{Im}(t) \\ \vec{cst} \end{pmatrix}.$$

Cependant, nous ne sommes plus dans le cadre d'un système dynamique linéaire. Nous ne pourrons plus avoir, en entrée du système, le vecteur d'état à l'instant précédent et, en sortie, le vecteur d'état à l'instant présent. Nous allons supposer que nous pouvons sortir de la matrice "variable" les facteurs de non-linéarité et les faire intervenir dans le vecteur d'entrée du système en espérant rendre la matrice constante dans le temps. Le but de cette démarche est de pouvoir conserver en sortie un "vecteur d'état". Notre système n'étant plus linéaire, nous continuons à appeler vecteur d'état, le vecteur qui sera utilisé à chaque étape pour calculer la sortie suivante. Nous conservons la dénomination "vecteur d'état" pour son sens intuitif. Remarquons que, si nous multiplions la constante d'entrée du modèle par un facteur non-linéaire, nous devrons aussi la multiplier dans le vecteur de sortie par ce même facteur sinon, nous n'aurons pas une matrice "constante".

$$\vec{x}_{IN}(t-h) = \vec{x}(t-h)\vec{\Upsilon}^T(t-h) = \begin{pmatrix} \alpha(t-h)\vec{p}_{Im}(t-h) \\ \beta(t-h)\vec{v}_{Im}(t-h) \\ \gamma(t-h)\vec{cst} \end{pmatrix}, \tag{6.5}$$

$$\vec{x}_{OUT}(t) = \vec{x}(t)\left(\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}\vec{\Upsilon}(t-h)\right)^T = \begin{pmatrix} \vec{p}_{Im}(t) \\ \vec{v}_{Im}(t) \\ \gamma(t-h)\vec{cst} \end{pmatrix}. \tag{6.6}$$

Nous aurions, ainsi, un système du type

$$\vec{x}_{OUT}(t) = \phi\vec{x}_{IN}(t).$$

Décomposons les vecteurs des équations 6.3 et 6.4 selon leurs différents composants, nous obtenons

$$p_{Im_x}(t) \stackrel{Def}{=} \frac{f}{p_z(t)}p_x(t),$$

$$p_{Im_y}(t) \stackrel{Def}{=} \frac{f}{p_z(t)}p_y(t)$$

et

$$v_{Im_x}(t) \stackrel{Def}{=} \frac{p_{Im_x}(t) - p_{Im_x}(t-h)}{h},$$

$$v_{Im_y}(t) \stackrel{Def}{=} \frac{p_{Im_y}(t) - p_{Im_y}(t-h)}{h}.$$

Désormais, nous avons la possibilité de reformuler le vecteur d'entrée en deux vecteurs et le vecteur de sortie, lui aussi, en deux vecteurs. Cela nous donne

$$\vec{x}_{IN_x}(t-h) = \begin{pmatrix} \alpha_x(t-h)p_{Im_x}(t-h) \\ \beta_x(t-h)v_{Im_x}(t-h) \\ \gamma_x(t-h)cst \end{pmatrix},$$

$$\vec{x}_{IN_y}(t-h) = \begin{pmatrix} \alpha_y(t-h)p_{Im_y}(t-h) \\ \beta_y(t-h)v_{Im_y}(t-h) \\ \gamma_y(t-h)cst \end{pmatrix},$$

$$\vec{x}_{OUT_x}(t) = \begin{pmatrix} p_{Im_x}(t) \\ v_{Im_x}(t) \\ \gamma_x(t-h)cst \end{pmatrix},$$

$$\vec{x}_{OUT_y}(t) = \begin{pmatrix} p_{Im_y}(t) \\ v_{Im_y}(t) \\ \gamma_y(t-h)cst \end{pmatrix}.$$

Cela permet d'écrire deux systèmes

$$\vec{x}_{OUT_x}(t) = \phi_x \vec{x}_{IN_x}(t),$$

$$\vec{x}_{OUT_y}(t) = \phi_y \vec{x}_{IN_y}(t).$$

Notre modèle neuro-mimétique aura pour objectif d'apprendre les matrices ϕ_x et ϕ_y. Pour atteindre ce but, nous allons, encore une fois, utiliser des réseaux mono-couche d'adalines. Chaque réseau sera en charge d'une des matrices. Pour chaque réseau d'adalines, l'ensemble d'apprentissage sera constitué des couples entrée-sortie à fournir à notre réseau d'adalines pour qu'il puisse réaliser son apprentissage. Nous avons comme ensemble d'apprentissage

$$\mathcal{A}_i = \{(\vec{x}_{IN_i}(t_0 + jh),\ \vec{x}_{OUT_i}(t_0 + (j+1)h)) \mid 0 \leq j \leq T\}$$

avec $i = x,y$, où i référence le réseau considéré pour l'apprentissage de la matrice ϕ_i. $T + 1$ est le nombre maximum d'exemples d'apprentissage. L'architecture du réseau, qui est chargé d'apprendre la matrice ϕ_i, est représentée à la figure 6.3.

La fonction d'apprentissage chargée de présenter successivement les couples d'entrée-

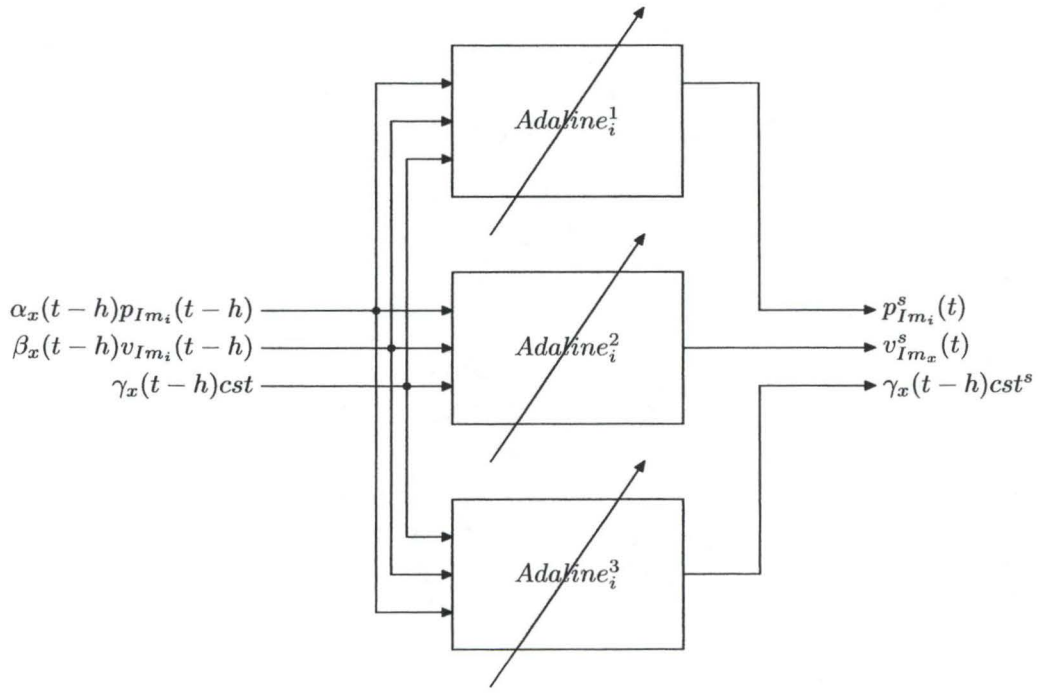


FIG. 6.3 – *Architecture des réseaux d'adalines pour l'apprentissage des matrices des systèmes dynamiques non-linéaires* ϕ_i.

sortie au réseau d'adalines i est représentée par

$$W_i^{new} = \mathcal{F}(\mathcal{A}_i, Tx, W_i)$$

où $\mathcal{F}$ est une fonction d'apprentissage définie au chapitre 2 (en pratique, nous prendrons $\mathcal{F} = \alpha$-LMS), $\mathcal{A}_i$ est un ensemble d'apprentissage, Tx est un taux d'apprentissage, W_i est une matrice des coefficients initiaux du réseau d'adalines et W_i^{new} est la matrice des coefficients du réseau d'adalines après apprentissage. Dans notre cas, les matrices W_i et W_i^{new} sont de la forme

$$\begin{pmatrix} w_{11}^i & w_{12}^i & w_{13}^i \\ w_{21}^i & w_{22}^i & w_{23}^i \\ w_{31}^i & w_{32}^i & w_{33}^i \end{pmatrix}.$$

Les coefficients w_{j1}^i sont les coefficients du premier adaline du réseau, w_{j2}^i ceux du deuxième et w_{j3}^i ceux du troisième. Ainsi, après apprentissage, nous avons

$$\vec{x}_{IN_i}(t_0 + jh)^T W_i^{new} = \vec{x}_i^s(t_0 + (j+1)h) \approx \vec{x}_{OUT_i}(t_0 + (j+1)h)^T \qquad (6.7)$$

où $\vec{x}_i^s(t_0 + (j+1)h)$ représente la sortie du réseau d'adalines. Nous n'avons pas l'égalité

entre $\vec{x}_i^s(t_0+(j+1)h)$ et $\vec{x}_{OUT_i}(t_0+(j+1)h)^T$ car il reste, après apprentissage, une erreur qui est négligeable.

Il reste un problème à résoudre: comment calculer le vecteur $\vec{\Upsilon}^T(t-h)$? Pour cela, nous allons nous baser sur la connaissance du modèle dans l'espace cartésien et du modèle de caméra "pin-hole". Nous allons identifier, non seulement les facteurs de non-linéarité mais aussi les matrices ϕ. Cela n'est pas gênant. En effet, nous aurons toujours besoin d'un apprentissage, car des paramètres propres à la disposition de la scène risquent d'intervenir dans les matrices ϕ. Si nous nous replaçons dans le contexte du laboratoire, ces paramètres sont supposés inconnus. En l'occurrence, nous ignorons la distance l qui sépare les deux caméras.

6.4 Calcul de $\vec{\Upsilon}^T(t-h)$

Reprenons les équations 6.3 et 6.4, nous avons

$$\vec{p}_{Im}(t) \stackrel{Def}{=} \frac{f}{p_z(t)}\vec{p}(t), \tag{6.8}$$

$$\vec{v}_{Im}(t) \stackrel{Def}{=} \frac{\vec{p}_{Im}(t)-\vec{p}_{Im}(t-h)}{h}. \tag{6.9}$$

Nous savons du chapitre 3 que

$$\vec{v}(t) = \frac{\vec{p}(t)-\vec{p}(t-h)}{h}+\frac{h}{2}\vec{g}, \tag{6.10}$$

$$\vec{p}(t) = \vec{p}(t-h)+\vec{v}(t-h)h+\frac{h^2}{2}\vec{g}. \tag{6.11}$$

En injectant 6.11 dans 6.8

$$\vec{p}_{Im}(t) = \frac{f}{p_z(t)}\vec{p}(t-h)+\frac{f}{p_z(t)}\vec{v}(t-h)h+\frac{f}{p_z(t)}\frac{h^2}{2}\vec{g}. \tag{6.12}$$

Nous pouvons exprimer $\vec{v}(t-h)$ en fonction de positions successives d'après l'équation 6.10. Cela nous donne

$$\vec{p}_{Im}(t) = \frac{f}{p_z(t)}\vec{p}(t-h)+\frac{f}{p_z(t)}\left(\frac{\vec{p}(t-h)-\vec{p}(t-2h)}{h}+\frac{h}{2}\vec{g}\right)h+\frac{f}{p_z(t)}\frac{h^2}{2}\vec{g}. \tag{6.13}$$

Distribuons et simplifions cette dernière équation et le résultat est:

$$\vec{p}_{Im}(t) = 2\frac{f}{p_z(t)}\vec{p}(t-h)-\frac{f}{p_z(t)}\vec{p}(t-2h)+\frac{f}{p_z(t)}\vec{g}h^2. \tag{6.14}$$

A partir de l'équation 6.8, exprimons $\vec{p}(t)$: cela nous fournit

$$\vec{p}(t) = \vec{p}_{Im}(t)\frac{p_z(t)}{f}. \tag{6.15}$$

Réexprimons alors l'équation 6.14 : nous avons

$$\vec{p}_{Im}(t) = 2\frac{f}{p_z(t)}\frac{p_z(t-h)}{f}\vec{p}_{Im}(t-h) - \frac{f}{p_z(t)}\frac{p_z(t-2h)}{f}\vec{p}_{Im}(t-2h) + \frac{fh^2}{p_z(t)}\vec{g}. \tag{6.16}$$

Reformulons cette dernière équation

$$\begin{aligned}\vec{p}_{Im}(t) &= \frac{p_z(t-2h)}{p_z(t)}\vec{p}_{Im}(t-h) - \frac{p_z(t-2h)}{p_z(t)}\vec{p}_{Im}(t-2h) \\ &\quad + \frac{2p_z(t-h) - p_z(t-2h)}{p_z(t)}\vec{p}_{Im}(t-h) + \frac{fh^2}{p_z(t)}\vec{g},\end{aligned} \tag{6.17}$$

$$\begin{aligned}&\overset{(6.9)}{=} \frac{p_z(t-2h)h}{p_z(t)}\vec{v}_{Im}(t-h) \\ &\quad + \frac{2p_z(t-h) - p_z(t-2h)}{p_z(t)}\vec{p}_{Im}(t-h) + \frac{fh^2}{p_z(t)}\vec{g}.\end{aligned} \tag{6.18}$$

Maintenant, appliquons le même schéma de raisonnement à partir de l'équation 6.9 :

$$\vec{v}_{Im}(t) = \frac{\vec{p}_{Im}(t) - \vec{p}_{Im}(t-h)}{h}, \tag{6.19}$$

$$\begin{aligned}&\overset{(6.16)}{=} \frac{p_z(t-2h)}{p_z(t)}\vec{v}_{Im}(t-h) + \frac{2p_z(t-h) - p_z(t-2h)}{p_z(t)h}\vec{p}_{Im}(t-h) \\ &\quad + \frac{fh}{p_z(t)}\vec{g} - \frac{\vec{p}_{Im}(t-h)}{h},\end{aligned} \tag{6.20}$$

$$\begin{aligned}&= \frac{2p_z(t-h) - p_z(t-2h) - p_z(t)}{p_z(t)h}\vec{p}_{Im}(t-h) + \frac{p_z(t-2h)}{p_z(t)}\vec{v}_{Im}(t-h) \\ &\quad + \frac{fh}{p_z(t)}\vec{g}.\end{aligned} \tag{6.21}$$

Nous pouvons exprimer les équations 6.18 et 6.21 sous la forme de notre modèle dynamique non-linéaire comme exprimé à l'équation 6.1. Cela nous donne en notation matricielle :

$$\begin{pmatrix} \vec{p}_{Im}(t) \\ \vec{v}_{Im}(t) \\ \vec{cst} \end{pmatrix} = \begin{pmatrix} \frac{2p_z(t-h)-p_z(t-2h)}{p_z(t)} & \frac{p_z(t-2h)h}{p_z(t)} & \frac{fh^2\vec{g}}{p_z(t)\vec{cst}} \\ \frac{2p_z(t-h)-p_z(t-2h)-p_z(t)}{hp_z(t)} & \frac{p_z(t-2h)}{p_z(t)} & \frac{fh\vec{g}}{p_z(t)\vec{cst}} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \vec{p}_{Im}(t-h) \\ \vec{v}_{Im}(t-h) \\ \vec{cst} \end{pmatrix}. \tag{6.22}$$

Or, nous savons du chapitre 3 que, si nous négligeons la force de frottement, nous

avons

$$p_z(t) = 2p_z(t-h) - p_z(t-2h) + g_z h^2.$$

Mais par hypothèse,

$$g_z = 0,$$

nous avons alors

$$p_z(t) = 2p_z(t-h) - p_z(t-2h). \tag{6.23}$$

En combinant les équations 6.22 et 6.23, nous obtenons comme modèle

$$\begin{pmatrix} \vec{p}_{Im}(t) \\ \vec{v}_{Im}(t) \\ \vec{cst} \end{pmatrix} = \begin{pmatrix} 1 & \frac{p_z(t-2h)h}{2p_z(t-h)-p_z(t-2h)} & \frac{fh^2}{2p_z(t-h)-p_z(t-2h)}\frac{\vec{g}}{\vec{cst}} \\ 0 & \frac{p_z(t-2h)}{2p_z(t-h)-p_z(t-2h)} & \frac{fh}{2p_z(t-h)-p_z(t-2h)}\frac{\vec{g}}{\vec{cst}} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \vec{p}_{Im}(t-h) \\ \vec{v}_{Im}(t-h) \\ \vec{cst} \end{pmatrix}. \tag{6.24}$$

Cela nous donne bien un modèle de la forme exprimée à l'équation 6.1.

La disparité à l'instant t vaut

$$\varrho(t) = \vec{p}_{Im_x}^{\,Caméra\ 1}(t) - \vec{p}_{Im_x}^{\,Caméra\ 2}(t).$$

Nous pouvons exprimer $p_z(t)$ comme nous l'avons vu au chapitre 4,

$$p_z(t) = \frac{lf}{\varrho(t)}$$

avec l l'écart entre les deux caméras. Nous pouvons, dès lors, reformuler notre modèle en fonction de la disparité :

$$\begin{pmatrix} \vec{p}_{Im}(t) \\ \vec{v}_{Im}(t) \\ \vec{cst} \end{pmatrix} = \begin{pmatrix} 1 & \frac{h\varrho(t-h)}{2\varrho(t-2h)-\varrho(t-h)} & \frac{h^2\varrho(t-h)\varrho(t-2h)}{l(2\varrho(t-2h)-\varrho(t-h))}\frac{\vec{g}}{\vec{cst}} \\ 0 & \frac{\varrho(t-h)}{2\varrho(t-2h)-\varrho(t-h)} & \frac{h\varrho(t-h)\varrho(t-2h)}{l(2\varrho(t-2h)-\varrho(t-h))}\frac{\vec{g}}{\vec{cst}} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \vec{p}_{Im}(t-h) \\ \vec{v}_{Im}(t-h) \\ \vec{cst} \end{pmatrix}. \tag{6.25}$$

Posons

$$\chi(t-h) = \frac{\varrho(t-h)}{2\varrho(t-2h) - \varrho(t-h)}.$$

Notre modèle devient

$$\begin{pmatrix} \vec{p}_{Im}(t) \\ \vec{v}_{Im}(t) \\ \vec{cst} \end{pmatrix} = \begin{pmatrix} 1 & h & \frac{h^2}{l}\frac{\vec{g}}{\vec{cst}} \\ 0 & 1 & \frac{h}{l}\frac{\vec{g}}{\vec{cst}} \\ 0 & 0 & \frac{1}{\varrho(t-2h)\chi(t-h)} \end{pmatrix} \begin{pmatrix} \vec{p}_{Im}(t-h) \\ \chi(t-h)\vec{v}_{Im}(t-h) \\ \varrho(t-2h)\chi(t-h)\vec{cst} \end{pmatrix}. \tag{6.26}$$

Nous voyons qu'il reste un terme non-linéaire dans la matrice. Or, ce terme est lié à la constante que nous avons fixée a priori. Nous pouvons le sortir, mais il faut le placer

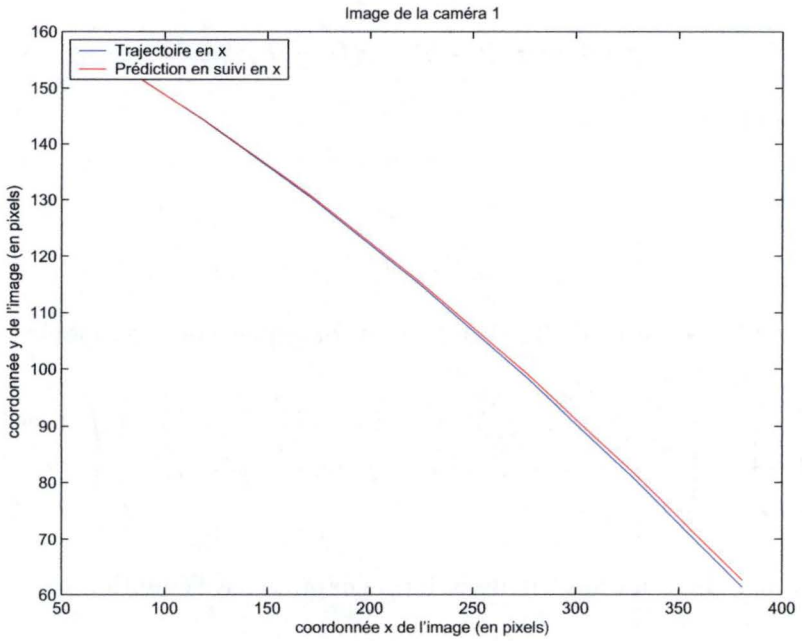


FIG. 6.4 – *Résultats de prédiction dans l'image.*

dans le vecteur de sortie, ainsi nous avons

$$\begin{pmatrix} \vec{p}_{Im}(t) \\ \vec{v}_{Im}(t) \\ \varrho(t-2h)\chi(t-h)\vec{cst} \end{pmatrix} = \begin{pmatrix} 1 & h & \frac{h^2}{l}\frac{\vec{g}}{\vec{cst}} \\ 0 & 1 & \frac{h}{l}\frac{\vec{g}}{\vec{cst}} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \vec{p}_{Im}(t-h) \\ \chi(t-h)\vec{v}_{Im}(t-h) \\ \varrho(t-2h)\chi(t-h)\vec{cst} \end{pmatrix}. \quad (6.27)$$

Nous avons identifié notre vecteur $\vec{\Upsilon}(t-h)$. Celui-ci vaut

$$\vec{\Upsilon}(t-h) = \begin{pmatrix} 1 \\ \chi(t-h) \\ \varrho(t-2h)\chi(t-h) \end{pmatrix}.$$

6.5 Validation du modèle avec données simulées avec le logiciel *Matlab*©

Les résultats sont présentés aux figures 6.4, 6.5 et 6.6. Ceux-ci sont exprimés en pixels. Nous avons utilisé un jeu d'apprentissage de 2000 trajectoires générées aléatoirement. Les caractéristiques de ces trajectoires sont semblables aux caractéristiques de celles réalisées, en réalité, au laboratoire. Une trajectoire de test est générée aléatoirement et constitue la source de notre mesure d'erreur.

A la figure 6.4, nous voyons, qu'après apprentissage, la prédiction à l'instant sui-

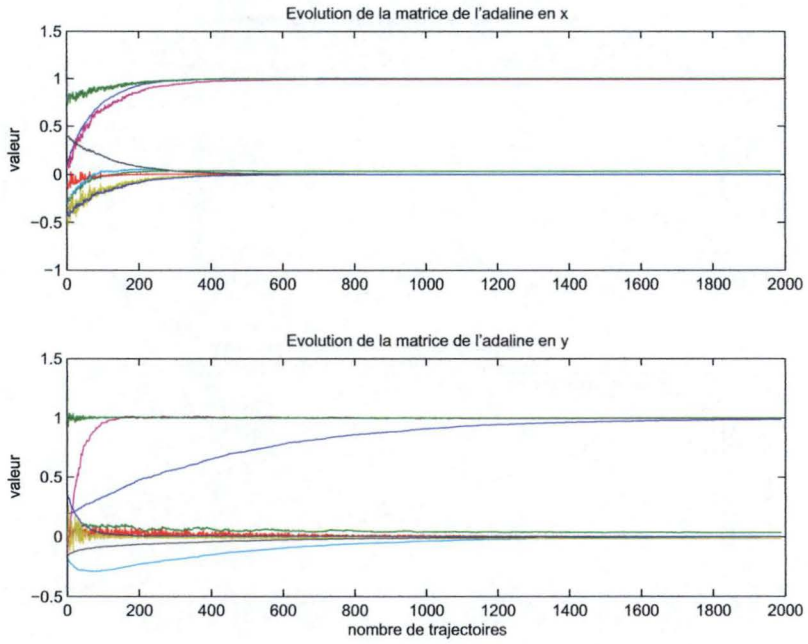


FIG. 6.5 – *Convergence des deux adalines.*

vant est excellente. Il est difficile de distinguer la trajectoire réelle de celle prédite.

L'erreur maximum mesurée (figure 6.6) est située aux alentours d'un pixel. Cette erreur est donc suffisante pour une exploitation réelle. Cependant, il ne faut pas oublier que cette erreur est une erreur à l'instant suivant et que nous travaillons sur des données simulées. La confrontation à la réalité, abordée à la section suivante, nous apprendra si notre modèle pourra être exploité.

La convergence des adalines se réalisent après environ 1400 trajectoires. Cette convergence est très lente. On peut, cependant, imaginer, en pratique, d'utiliser un certain nombre de trajectoires aléatoires et ensuite d'utiliser des trajectoires réelles pour faire converger le modèle. De plus, nous pourrions optimiser les entrées et sorties du modèle afin de le faire converger plus vite.

Les coefficients des matrices responsables de la prédiction dans une des images valent

$$\phi_x^{Expérience} = \begin{pmatrix} 1.0001 & 0.0331 & -0.0001 \\ 0.0021 & 0.9916 & -0.0022 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix} \approx \phi_x,$$

$$\phi_y^{Expérience} = \begin{pmatrix} 0.9975 & 0.0333 & -0.0008 \\ -0.0013 & 0.9953 & 0.0176 \\ 0.0002 & 0.0000 & 1.0000 \end{pmatrix} \approx \phi_y.$$

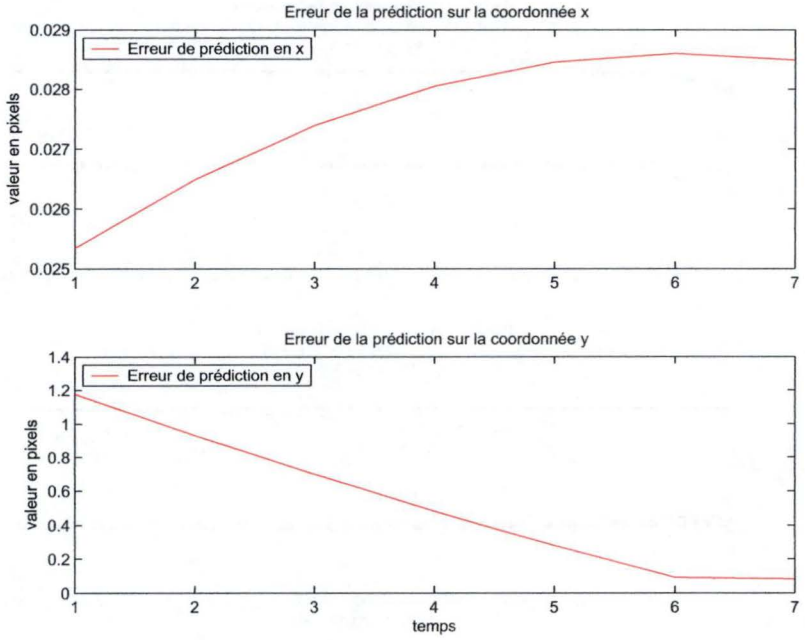


FIG. 6.6 – *Erreur de prédiction en pixels.*

Nous voyons qu'ils tendent bien vers les coefficients théoriques que nous avons établis précédemment.

6.6 Résultats sur des données réelles

Nous présentons les résultats aux figures 6.7, 6.8 et 6.9. Les résultats sont mesurés en pixels. Le jeu d'apprentissage utilisé est de 1000 trajectoires simulées générées aléatoirement et suivies de 53 trajectoires réelles présentées 100 fois. Le taux d'apprentissage est de 0.9 pour les trajectoires simulées. Pour les trajectoires réelles, celui-ci décroît dans le temps. Les caractéristiques des trajectoires simulées sont semblables aux caractéristiques de celles réalisées, en réalité, au laboratoire. Une trajectoire réelle constitue la source de notre mesure d'erreur. Nous voyons que les erreurs (figure 6.9) sont beaucoup trop importantes pour espérer une utilisation à long terme, c'est-à-dire réaliser une prédiction à partir des prédictions précédentes en itérant.

6.7 Conclusion

Le modèle développé dans ce chapitre répond, en partie, à l'objectif d'identification du système. En effet, il apprend des coefficients dépendants de la gravité, de la distance entre les caméras et de la focale. Mais du fait que le modèle dans l'image est devenu

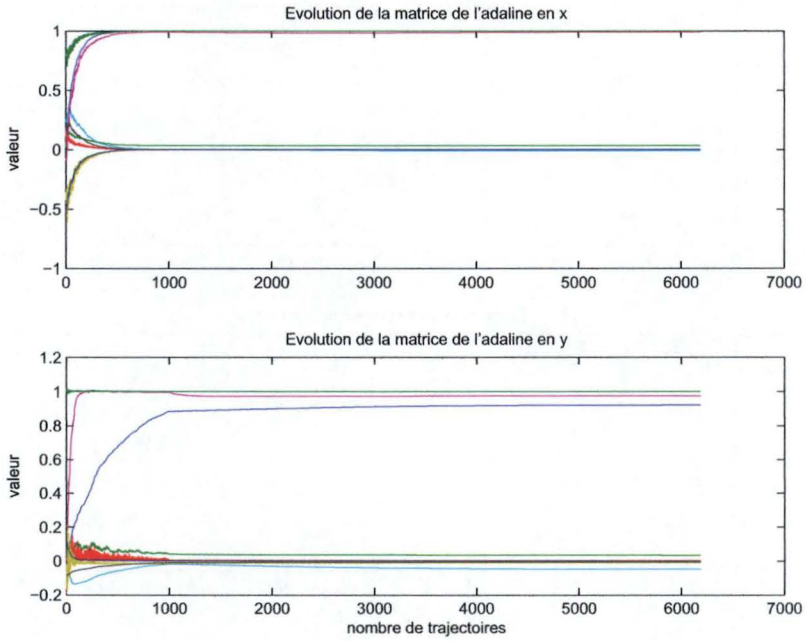


FIG. 6.7 – *Convergence des adalines sur un ensemble de 1000 trajectoires simulées, générées aléatoirement et suivies de 53 trajectoires réelles présentées 100 fois. Le taux d'apprentissage est décroissant à partir des trajectoire réelles.*

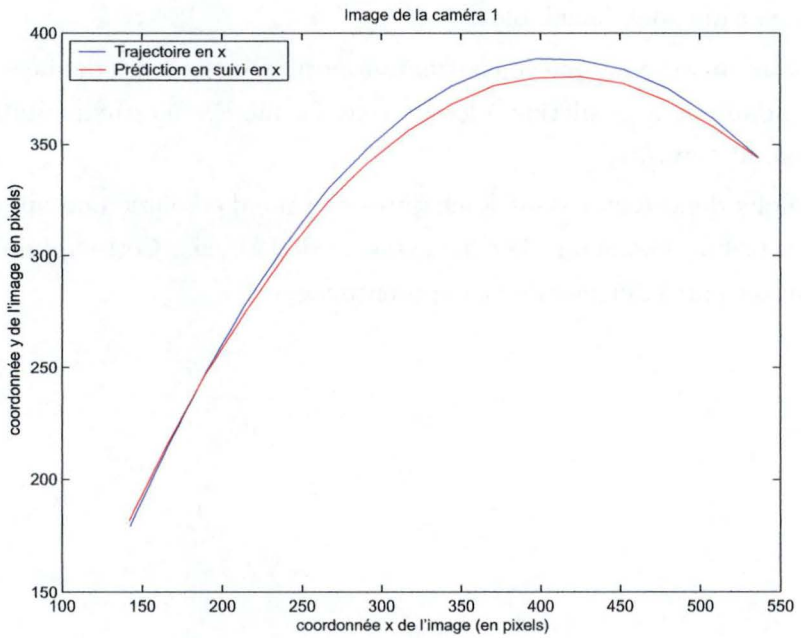


FIG. 6.8 – *Résultats de prédiction dans l'image sur une trajectoire réelle.*

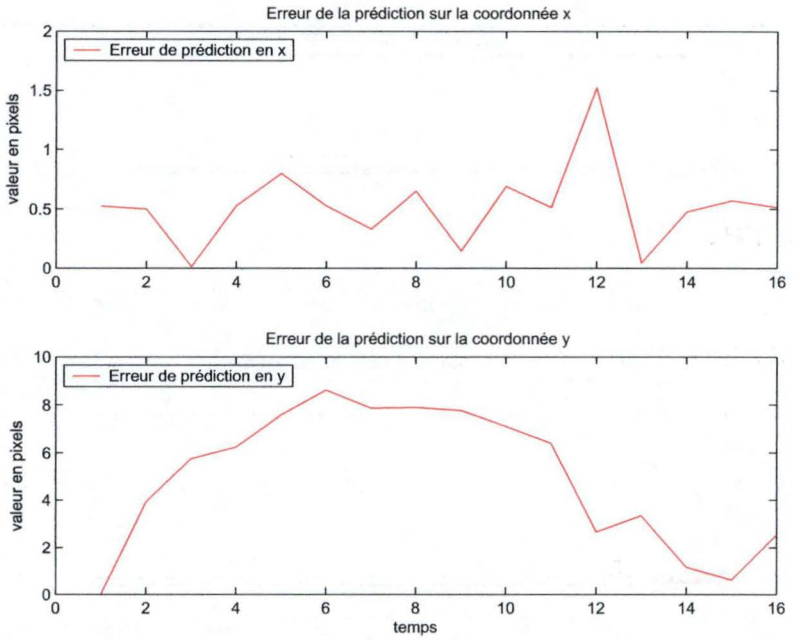


FIG. 6.9 – *Erreur de prédiction en pixels à l'instant suivant sur une trajectoire réelle. Ces erreurs sont beaucoup trop importantes pour espérer une utilisation à long terme.*

non-linéaire, il n'est plus possible d'identifier complètement ce système uniquement avec des adalines. Nous avons donc dû extraire les coefficients qui entraînent une non-linéarité et qui sont calculables.

Nous avons aussi remarqué en confrontant notre modèle à des données réelles qu'il n'était pas adapté à la prédiction à long terme. Ce modèle n'est exploitable que pour prédire l'instant suivant.

Nous allons donc tenter dans le chapitre suivant d'élaborer une approche mixte en tirant parti d'un maximum de connaissances de la scène. Cette approche sera une approche située entre l'ingénierie et l'apprentissage.

Chapitre 7

Modèle de prédiction balistique - Caméras non-parallèles

7.1 Introduction

Nous avons vu au chapitre précédent que les données réelles posaient un problème pour prédire à long terme. Ce chapitre change d'approche et va tirer parti de la connaissance de la fonction de profondeur décrite au chapitre 4 et développée en annexe à la page 143. Nous allons toujours essayer de rester dans le cadre de l'adaline ou d'une méthode équivalente.

Ce chapitre présente un double but. Le premier est de lever l'hypothèse des caméras parallèles et d'établir ainsi un modèle pour des caméras dont les axes focaux ne sont plus parallèles. Les caméras sont toujours situées dans un même plan.

Le second but visé est d'évaluer si l'approche que nous allons développer, peut être utilisée, en pratique, avec des données réelles qui présentent un manque de précision par rapport aux données simulées.

7.2 Hypothèses

Cette approche ne considère qu'une seule véritable hypothèse : la force de frottement n'est pas très importante sur le mobile. Grâce à cette hypothèse, nous allons pouvoir raisonner indépendamment de toute force de frottement même si, en simulation, nous allons les considérer.

Les caméras ne sont pas, forcément, parallèles. Par contre, par facilité de développement théorique, nous considérons que la gravité est perpendiculaire au plan contenant les axes optiques. Cette hypothèse est utilisée par commodité. La lever est très simple et ne modifie que les développements théoriques de la deuxième phase de la méthode. La configuration de la scène est illustrée par les figures 7.1 et 7.2.

7.3 Méthode

La méthode présentée, dans ce chapitre, se décompose en deux phases (figure 7.3). La première phase consiste à apprendre la fonction de profondeur de la caméra 1.

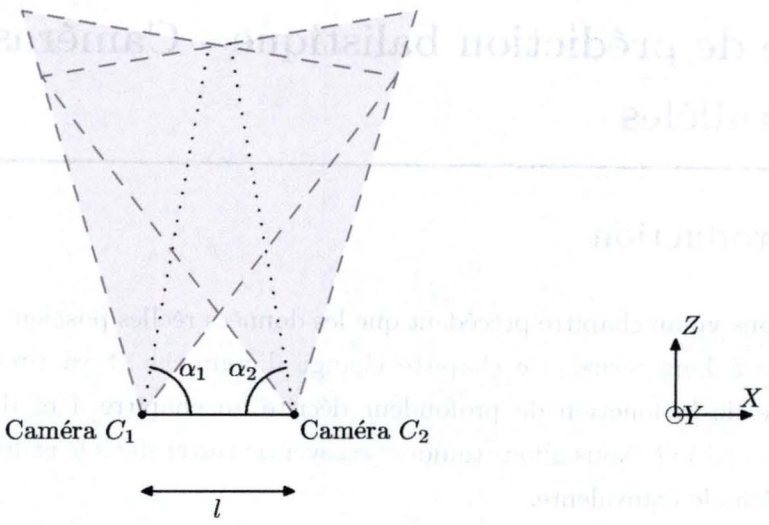


FIG. 7.1 – *Vue du dessus de la scène.*

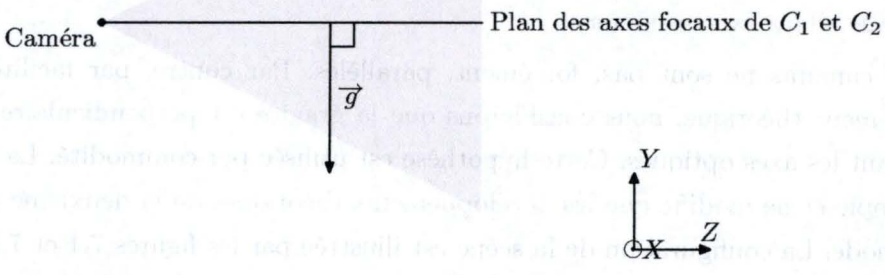


FIG. 7.2 – *Vue latérale de la scène.*

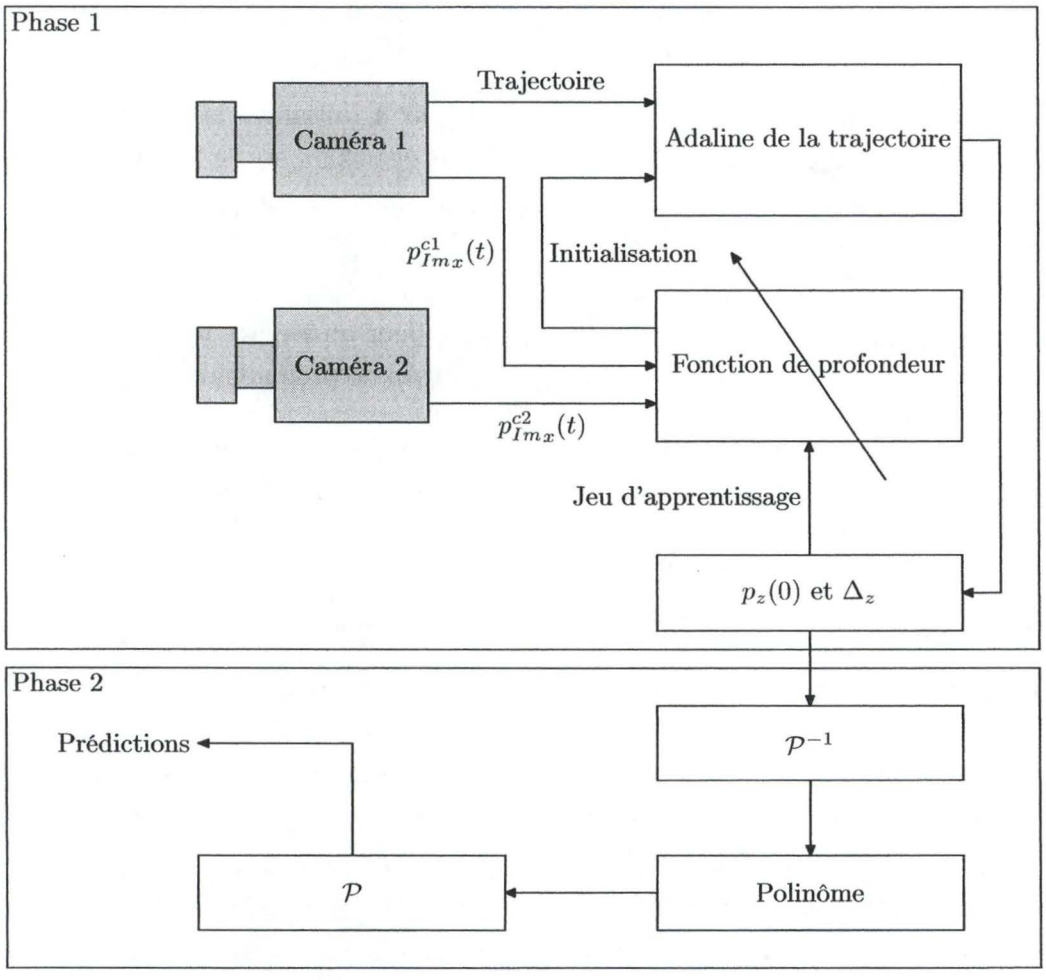


FIG. 7.3 – *Méthode de prédiction dans l'image de la caméra 1.*

Cette phase est une phase de calibration "automatique". Cette fonction de profondeur, développée à l'annexe C, nécessite de connaître les coordonnées en abscisse des deux caméras, du point dont on veut déterminer la profondeur. Cette fonction est de la forme

$$p_z(t) = \frac{-Ap_{Imx1}(t) + B}{(p_{Imx2}(t) - p_{Imx1}(t))C - Dp_{Imx1}(t)p_{Imx2}(t) - E}. \tag{7.1}$$

Les coefficients de cette fonction sont déterminés, au départ, de manière arbitraire. Nous pouvons, par exemple, considérer que les caméras sont parallèles, même si elles ne le sont pas, et déterminer ainsi, les coefficients de départ.

Lors d'une trajectoire, cette fonction va servir à initialiser les coefficients d'un adaline propre à cette trajectoire. Le but de cette opération est de faire converger cet adaline plus rapidement. Cet adaline va nous permettre de trouver la profondeur du premier point par rapport à une des deux caméras ainsi que la variation constante de profondeur entre deux points successifs de la trajectoire. Grâce à ces données, nous serons en mesure de créer un jeu d'apprentissage pour un second adaline dont le rôle est d'adapter les coefficients de la fonction théorique de profondeur (équation 7.1).

Après apprentissage de la fonction, le rôle de l'adaline propre à la trajectoire est diminué. Il constitue davantage une sorte de filtre qui diminue l'erreur de mesure. Nous pouvons donc directement exploiter la première mesure de profondeur et de variation de profondeur.

La seconde phase consiste à corriger l'effet de la perspective grâce à la fonction inverse $\mathcal{P}^{-1}$ de la projection perspective. Après cette correction, nous nous retrouvons dans un espace cartésien et nous pouvons prédire la trajectoire avec un polynôme de degré deux pour l'axe faisant intervenir la gravité et un polynôme de degré un pour l'autre (si les caméras sont placées de la sorte). Une fois la prédiction faite, nous pouvons repasser dans l'image avec la fonction de projection perspective $\mathcal{P}$. Le fait d'utiliser un polynôme au lieu du modèle du chapitre 3 est pure commodité. Nous aurions pu utiliser le modèle du chapitre 3, mais la mise en oeuvre dans les simulations était plus longue alors que $Matlab_{©}$ fournit des outils rapides pour les polynômes.

7.4 Adaline propre à une trajectoire

Considérons l'équation de la trajectoire comme présentée au chapitre 3 et ne considérons, dans un premier temps, que l'axe Y faisant intervenir la gravité. Cette équation peut être reformulée pour ne dépendre que de positions successives du mobile et de la gravité.

$$2p_y(t) - p_y(t-h) + gh^2 = p_y(t+h). \tag{7.2}$$

Or, dans le chapitre 4, nous avons présenté le modèle de projection caméra comme

$$p_{Im_y}(t) = \frac{p_y(t)f}{p_z(t)\psi_y} - \frac{v_{max}}{2}. \tag{7.3}$$

Notons que

$$\mathcal{P}(p_y(t),p_z(t)) = \frac{p_y(t)f}{p_z(t)\psi_y} - \frac{v_{max}}{2}.$$

De l'équation 7.3, nous allons exprimer $p_y(t)$ en fonction de $p_{Im_y}(t)$, cela nous donne

$$p_y(t) = (p_{Im_y}(t) + \frac{v_{max}}{2})\frac{(p_z(t)\psi_y)}{f}. \tag{7.4}$$

Notons que

$$p_y(t) = \mathcal{P}^{-1}(p_{Im_y}(t),p_z(t)).$$

Utilisons l'expression de $p_y(t)$ de l'équation 7.4 dans l'équation 7.2, nous avons

$$\begin{aligned} &2(p_{Im_y}(t) + \frac{v_{max}}{2})(\frac{p_z(t)\psi_y}{f}) - (p_{Im_y}(t-h) + \frac{v_{max}}{2})(\frac{(p_z(t)-\Delta_z)\psi_y}{f}) \\ &-(p_{Im_y}(t+h) + \frac{v_{max}}{2})(\frac{(p_z(t)+\Delta_z)\psi_y}{f}) = -gh^2 \end{aligned}$$

où Δ_z vaut $p_z(t) - p_z(t-h)$. Le membre ψ_y est présent dans chaque terme de la partie droite de l'équation ; divisons alors l'équation par ψ_y

$$2(p_{Im_y}(t)+\frac{v_{max}}{2})(\frac{p_z(t)}{f})-(p_{Im_y}(t-h)+\frac{v_{max}}{2})(\frac{(p_z(t)-\Delta_z)}{f})-(p_{Im_y}(t+h)+\frac{v_{max}}{2})(\frac{(p_z(t)+\Delta_z)}{f}) = \frac{-gh^2}{\psi_y}.$$

Distribuons et simplifions

$$2p_{Im_y}(t)p_z(t) - p_{Im_y}(t-h)p_z(t) + p_{Im_y}(t-h)\Delta_z - p_{Im_y}(t+h)p_z(t) - p_{Im_y}(t+h)\Delta_z = \frac{-gfh^2}{\psi_y}.$$

Cette dernière équation peut être exprimée de manière matricielle afin de pouvoir être apprise par un adaline

$$\begin{pmatrix} 2p_{Im_y}(t) - p_{Im_y}(t-h) - p_{Im_y}(t+h) \\ p_{Im_y}(t-h) - p_{Im_y}(t+h) \end{pmatrix}^T \begin{pmatrix} p_z(t) \\ \Delta_z \end{pmatrix} = \frac{-gfh^2}{\psi_y}. \tag{7.5}$$

Dès lors, nous disposons d'un adaline dont les coefficients, après convergence, sont la profondeur et la variation de profondeur. Grâce à ces données, nous sommes en mesure de corriger la déformation due à la perspective pour la trajectoire considérée. Remarquons encore que l'on peut faire converger cet adaline avec une autre valeur que $\frac{-gfh^2}{\psi_y}$. Dans ce cas, nous obtenons la profondeur et la variation de celle-ci à un facteur d'échelle près. Cela n'est pas contrariant pour les développements futurs ni

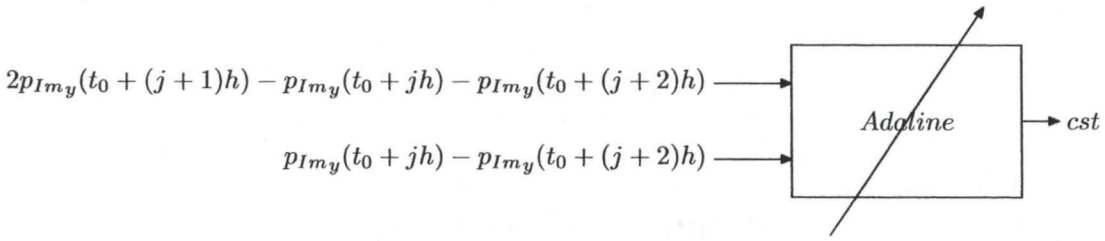


FIG. 7.4 – *Adaline propre à une trajectoire.*

pour la correction de la perspective.

L'ensemble d'apprentissage de l'adaline sera

$$\vec{IN}(t_0,j) = \begin{pmatrix} 2p_{Im_y}(t_0+(j+1)h) - p_{Im_y}(t_0+jh) - p_{Im_y}(t_0+(j+2)h) \\ p_{Im_y}(t_0+jh) - p_{Im_y}(t_0+(j+2)h) \end{pmatrix},$$

$$\mathcal{A} = \left\{ \left(\vec{IN}(t_0,j), cst \right) \mid 0 \leq j \leq T-2 \right\}$$

où $T-1$ est la taille de l'ensemble d'apprentissage et t_0 l'instant initial. L'adaline responsable de l'apprentissage des coefficients $p_z(t) et\ \Delta_z$, à l'échelle près, est représenté par la figure 7.4.

7.5 Adaline utile à l'adaptation des coefficients de la fonction théorique de profondeur

A partir de $p_z(t)$ et de Δ_z, on peut recréer les $p_z(t) \ldots p_z(t+kh)$ correspondant aux profondeurs suscessives des points de la trajectoire. D'autre part, dans le chapitre 4, nous avons décrit une fonction capable de nous donner la profondeur d'un point perçu dans une caméra grâce aux abscisses de ce point dans les deux caméras. Cette fonction est valable quelle que soit la position des caméras. Nous avons, donc,

$$p_z(t) = \frac{-Ap_{Im_{x1}}(t) + B}{(p_{Im_{x2}}(t) - p_{Im_{x1}}(t))C - Dp_{Im_{x1}}(t)p_{Im_{x2}}(t) - E}.$$

En multipliant le membre de gauche par le dénominateur de la fraction, nous avons

$$p_z(t)(p_{Im_{x2}}(t) - p_{Im_{x1}}(t))C - Dp_z(t)p_{Im_{x1}}(t)p_{Im_{x2}}(t) - p_z(t)E = -Ap_{Im_{x1}}(t) + B.$$

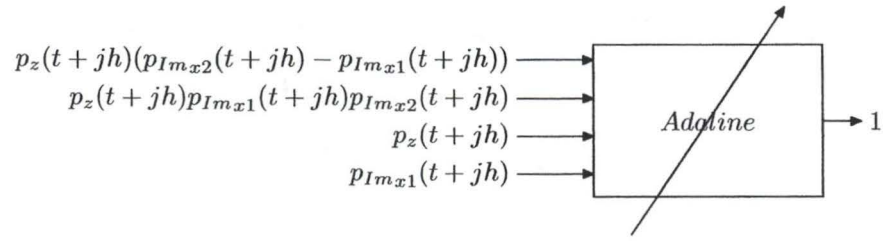


FIG. 7.5 – *Adaline pour apprendre les coefficients de la fonction de profondeur, à l'échelle près.*

Mettons tous les termes faisant intervenir des points images des caméras dans le membre de gauche

$$p_z(t)(p_{Im_{x2}}(t) - p_{Im_{x1}}(t))C - Dp_z(t)p_{Im_{x1}}(t)p_{Im_{x2}}(t) - p_z(t)E + Ap_{Im_{x1}}(t) = B.$$

Exprimons cette équation de manière à faire apprendre les coefficients inconnus par un adaline

$$\begin{pmatrix} p_z(t)(p_{Im_{x2}}(t) - p_{Im_{x1}}(t)) \\ p_z(t)p_{Im_{x1}}(t)p_{Im_{x2}}(t) \\ p_z(t) \\ p_{Im_{x1}}(t) \end{pmatrix}^T \begin{pmatrix} C \\ -D \\ -E \\ A \end{pmatrix} = B.$$

Divisons tous les coefficients par B,

$$\begin{pmatrix} p_z(t)(p_{Im_{x2}}(t) - p_{Im_{x1}}(t)) \\ p_z(t)p_{Im_{x1}}(t)p_{Im_{x2}}(t) \\ p_z(t) \\ p_{Im_{x1}}(t) \end{pmatrix}^T \begin{pmatrix} \frac{C}{B} \\ -\frac{D}{B} \\ -\frac{E}{B} \\ \frac{A}{B} \end{pmatrix} = 1.$$

L'ensemble d'apprentissage de l'adaline sera

$$\vec{IN}(t_0,j) = \begin{pmatrix} p_z(t+jh)(p_{Im_{x2}}(t+jh) - p_{Im_{x1}}(t+jh)) \\ p_z(t+jh)p_{Im_{x1}}(t+jh)p_{Im_{x2}}(t+jh) \\ p_z(t+jh) \\ p_{Im_{x1}}(t+jh) \end{pmatrix},$$

$$\mathcal{B} = \left\{ \left(\vec{IN}(t_0,j), 1 \right) \mid 0 \leq j \leq T \right\}$$

où $T+1$ est la taille de l'ensemble d'apprentissage et t_0 l'instant initial. L'adaline responsable de l'apprentissage des coefficients de la fonction théorique, à l'échelle près, est représenté par la figure 7.5.

A l'aide de deux points successifs et de la fonction ainsi déterminée, on peut effectuer la correction de l'effet de la perspective grâce à

$$\mathcal{P}^{-1}(p_{Im_y}(t),p_z(t)) = \frac{(p_{Im_y}(t) + \frac{v_{max}}{2})p_z(t)\psi_y}{f} = p_y(t).$$

7.6 Prédiction à l'aide d'un polynôme

Pour pouvoir prédire à long terme, nous allons utiliser un polynôme de degré deux qui possède comme variable le temps. En effet, notre trajectoire dans l'espace cartésien, après correction de la perspective, peut s'écrire comme

$$p_y(t) = p_y(0) + v_y(0)t + \frac{gt^2}{2}$$

et

$$p_x(t) = p_x(0) + v_x(0)t.$$

Remarquons que, si nous avons effectué la correction de la perspective avec une profondeur et une variation de profondeur qui possèdent un facteur d'échelle, alors nous retrouvons ces polynômes mais avec des coefficients différents.

Trois points nous permettent déjà d'avoir une première estimation de la trajectoire au cours du temps. Le fait de nécessiter seulement trois points est un avantage. En effet, le but final est de pouvoir éventuellement intercepter le mobile. Dès lors, plus vite on peut bénéficier d'une approximation du point d'interception, plus on augmente les chances d'arriver à placer l'effecteur du bras robotique à temps pour cette interception.

7.7 Dispositions pratiques des simulations

Pour réaliser nos simulations, nous allons séparer notre méthode en deux phases. La première phase, nous permettra de découvrir les coefficients exacts de la fonction de profondeur. A cette fin, nous effectuerons une régression linéaire en lieu et place de l'adaline de la trajectoire. Cela revient à un adaline en mode "batch". Nous accomplissons cela dans un souci de rapidité, car l'adaline en ligne peut prendre plus de temps à converger. Suite à cela, nous utiliserons un adaline en "mode batch" pour apprendre les coefficients de la fonction. En effet, lors de nos expérimentations, nous avons vu qu'un adaline en ligne mettait trop de temps à converger alors qu'en "batch", une vingtaine de trajectoires suffit à obtenir les coefficients de manière assez précise.

Une fois les coefficients de la fonction appris, nous passons à la phase deux. Lors de cette phase, nous utiliserons la fonction théorique de profondeur pour évaluer la

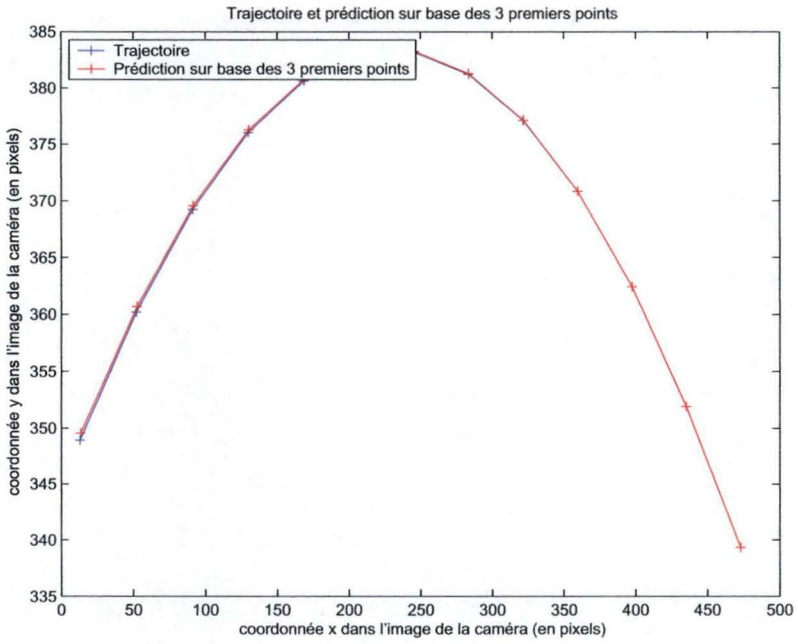


FIG. 7.6 – *Prédiction dans l'image à partir de 3 points.*

profondeur d'un point.

7.8 Résultats des simulations

Nous utilisons 20 trajectoires générées aléatoirement selon des conditions proches du cadre d'expérimentation pour trouver les coefficients de la fonction de profondeur. Les trajectoires sont générées en tenant compte de la force de frottement. Après cela, nous créons une nouvelle trajectoire aléatoire pour mesurer l'erreur de prédiction dans l'image. Les polynômes représentant la trajectoire sont calculés sur 3 points consécutifs de celle-ci.

La figure 7.6 montre la prédiction dans l'image. La figure 7.7 correspond aux erreurs sur les deux axes. Nous voyons que l'erreur à la fin de la trajectoire est voisine d'un pixel au maximum. Ces résultats sont très bons étant donné que,à l'aide des trois premiers points de la trajectoire, nous arrivons à être précis au pixel prés.

Au niveau de l'apprentissage, nous voyons qu'en séparant les deux phases et en utilisant le mode "batch", nous arrivons à trouver nos coefficients très rapidement. Nous aurions pu utiliser le mode "en ligne", mais dans ce cas un ensemble d'apprentissage bien plus grand aurait dû être utilisé. Il faut donc seulement une vingtaine de trajectoires pour effectuer la phase de calibration. De plus, avec trois points d'une trajectoire, nous arrivons à avoir une prédiction à long terme très précise. Nous ver-

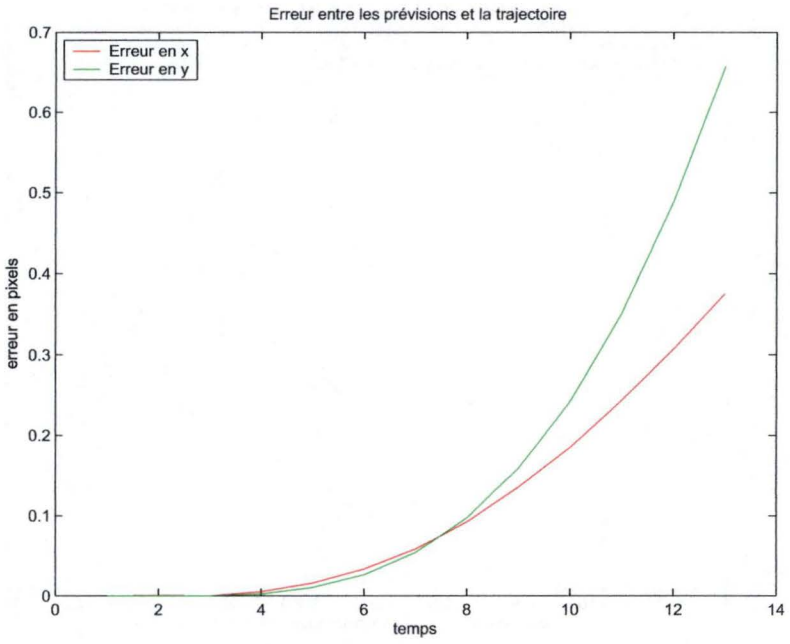


FIG. 7.7 – *Erreur de prédiction en pixels.*

rons dans la section suivante si cette méthode est réaliste. En effet, l'acquisition de données produit un bruit de mesure d'au moins un à deux pixels.

7.9 Analyse de sensibilité au bruit de la fonction de profondeur et de l'adaline propre à la trajectoire

Pour déterminer si notre démarche est réaliste, nous allons considérer la fonction théorique de profondeur et voir si un bruit aléatoire de un à deux pixels a ou non, un trop grand impact sur le calcul de la profondeur. Nous allons considérer qu'une des deux caméras forme un angle de 5 degrés par rapport à la droite parallèle à l'axe focal de l'autre caméra. Nous allons considérer un point quelconque et créer d'autres points à partir de celui-ci en ajoutant un bruit jusqu'à deux pixels au maximum sur l'abscisse d'une des deux caméras.

La figure 7.8 présente les résultats de l'analyse. Nous voyons que la profondeur peut varier jusqu'à 30 cm en fonction du bruit. Cette variation est trop grande pour pouvoir exploiter cette approche de manière correcte, d'autant plus que le bruit a une distribution dont nous ne connaissons aucune propriété. En effet, il est dû, en partie à la pixélisation et en partie, à la désynchronisation des caméras et au traitement d'images en général.

L'adaline, propre à une trajectoire, ne converge plus vers les valeurs correctes

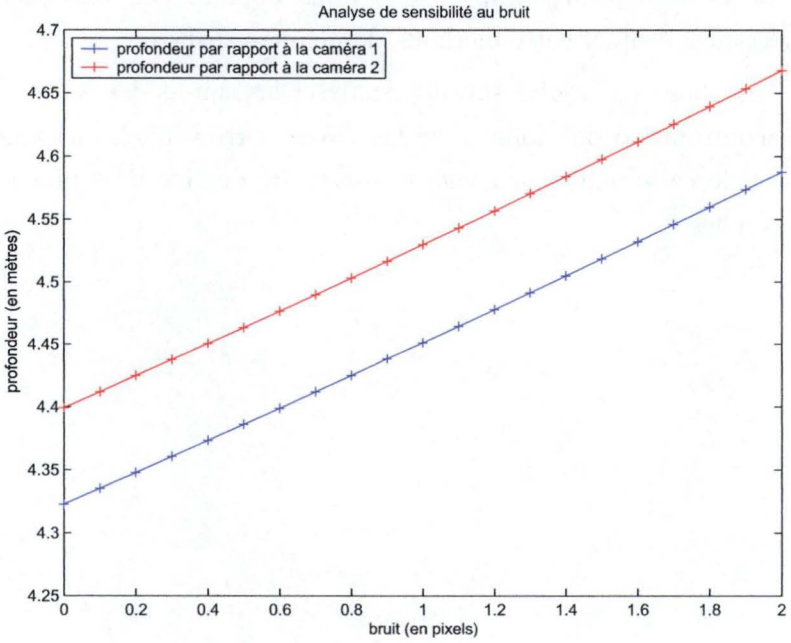


FIG. 7.8 – *Analyse de sensibilité.*

de profondeur et de variation de profondeur. Le bruit de deux pixels dégrade son apprentissage. La méthode présentée dans ce chapitre semble donc inappropriée au contexte réel d'application.

7.10 Conclusion

L'approche présentée dans ce chapitre paraît séduisante sur le plan théorique et en simulation. En effet, elle permet d'effectuer un calibrage automatique alors que d'autres systèmes nécessitent des calibrations par grille d'étalonnage. D'autre part, avec une vingtaine de trajectoires de calibration, nous disposons d'assez de précision pour corriger l'impact de la perspective et prédire à long terme avec seulement trois points d'une trajectoire.

Cette approche répond au premier objectif fixé par ce chapitre : savoir prédire avec des caméras qui ne sont plus parallèles. Nous sommes donc dans un cadre de prédiction beaucoup plus général.

Cependant, nous avons vu que cette méthode est lente à converger en ligne. Il faudrait donc la séparer en une phase de calibration à part entière et une phase de prédiction. Ensuite, nous avons vu que la fonction de profondeur ainsi découverte et l'adaline propre à la trajectoire sont beaucoup trop sensibles au bruit, ce qui rend

cette approche inutilisable en pratique. Nous avons répondu, par conséquent, au second objectif qui visait à évaluer cette méthode de manière réaliste.

Nous allons, dans le chapitre suivant, analyser les causes des échecs successifs de nos modèles confrontés à des données réelles. Nous verrons quels enseignements nous pourrons tirer de cette analyse en vue de construire des modèles plus efficaces avec des données réelles.

Chapitre 8

Analyse des sources des problèmes

8.1 Introduction

Les deux modèles, présentés respectivement dans les chapitres 6 et 7, ont été invalidés lors de leur confrontation avec des données réelles. Ce chapitre va analyser quelles sont les sources de l'invalidation de ces modèles. Pour cela, nous aborderons deux grands points.

Le premier consistera à déterminer si, dans une acquisition d'images parfaite, la pixélisation peut avoir un impact important sur nos modèles. Nous verrons si cette pixélisation peut être gênante dans la détermination du centre de gravité de la balle. Ensuite, nous étudierons ce même problème dans le cadre d'acquisition par les webcams utilisées pour collecter des données réelles.

En second lieu, nous analyserons l'acquisition de données par deux webcams et nous verrons si notre mécanisme de synchronisation est réellement efficace. Dans le cas contraire, nous déterminerons l'impact engendré sur la construction de modèles.

De ces deux grands points, découlera un certain nombre de conséquences qui nous seront utiles pour élaborer de nouveaux modèles efficaces avec des données réelles. Nous constaterons aussi si les outils utilisés sont assez précis pour répondre au but recherché.

8.2 Bruit de pixélisation et de traitement d'images

Le bruit de pixélisation est provoqué par la discrétisation en pixel d'un objet et par le traitement d'images qui tente d'extraire la balle de l'ensemble de l'image. Ce bruit peut être plus ou moins important. Pour s'en convaincre intuitivement, imaginons une balle observée par une caméra sur un fond uniformément noir. Cette balle peut être proche de la caméra. En pareil cas, elle sera finement discrétisée par un grand nombre de pixels (figure 8.1). Maintenant, éloignons la balle de la caméra. Seulement quelques pixels représentent la balle (figure 8.2). Cela correspond au premier aspect du bruit de discrétisation.

Un second aspect consiste à considérer l'acquisition de la balle par une caméra. Pour pouvoir déterminer un centre de gravité, le traitement d'images nécessitera de définir un seuil en dessous duquel les pixels sont ignorés. En fonction de beaucoup de

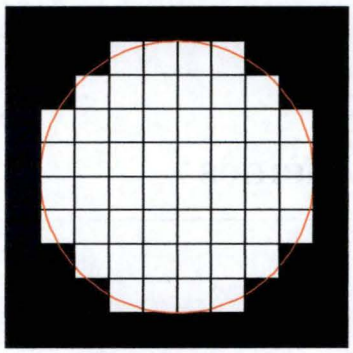


FIG. 8.1 – *Dicrétisation par pixélisation et seuillage d'une balle proche de la caméra.*

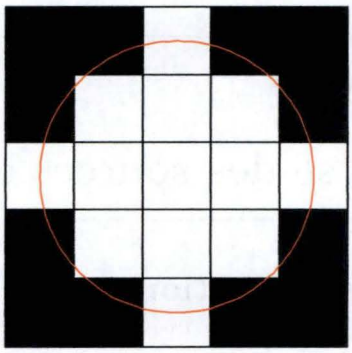


FIG. 8.2 – *Dicrétisation par pixélisation et seuillage d'une balle éloignée de la caméra.*

facteurs tels que la luminosité, les réglages de la caméra, etc., la balle sera représentée par des pixels d'intensité différente. Nous aurons donc une approximation de ce centre de gravité. En effet, certains pixels d'intensité faible seront ignorés à cause du seuillage alors qu'ils auraient été nécessaires pour calculer un centre de gravité plus proche du centre de gravité réel. Cette situation est illustrée à la figure 8.3. Dans cette figure, nous voyons que le contour de la balle extrait de l'image possède un centre de gravité plus bas que celui de la balle. Cela constitue une première source de bruit sur le centre de gravité.

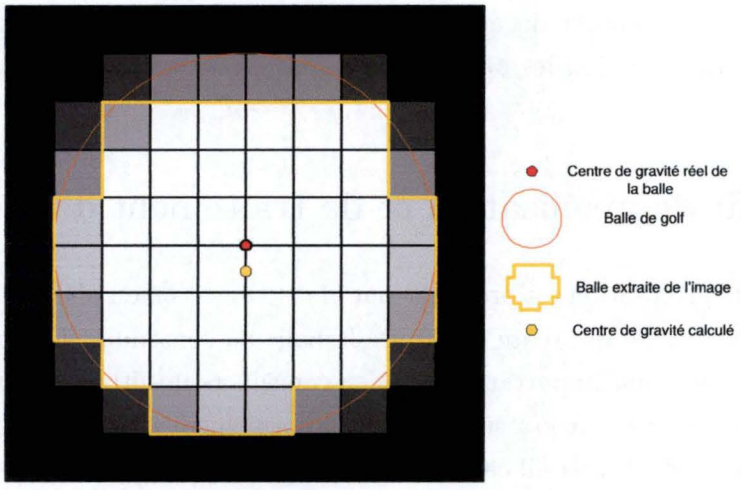


FIG. 8.3 – *Erreur de mesure du centre de gravité.*

Un troisième aspect qui intervient dans le bruit de discrétisation est celui du fonctionnement du matériel utilisé. Nous avons réalisé nos simulations avec des webcams qui intègrent l'image pendant une période donnée. Cela veut dire que ces webcams

enregistrent l'image sur une période de temps t_i. Cette donnée est inconnue. Cela a pour conséquence que la balle observée à un instant t_0+kh est en fait la balle observée pendant l'intervalle de temps $t_0+kh-t_i-\delta$ et $t_0+kh-\delta$ (figure 8.4). Le δ représente le temps entre la fin de l'observation et la mise à la disposition de cette image. Cet intervalle de temps est aussi inconnu. Le fait d'avoir une intégration longue implique que la balle observée sur l'image n'est pas ronde mais forme une traînée (figure 8.5). Cette traînée est légèrement courbe. En effet, elle constitue une partie de la trajectoire. Le fait de ne connaître ni t_i ni δ rend l'erreur de mesure du centre de gravité de la balle à un instant donné très variable. Si nous supposons que ces deux valeurs sont assez constantes, l'erreur engendrée lors du calcul du centre de gravité peut rester raisonnable.

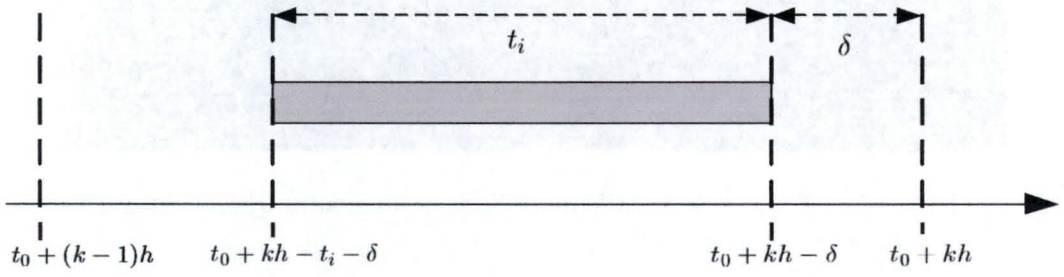


FIG. 8.4 – *Fonctionnement de l'intégration d'une image provenant d'une webcam.*

8.3 Désynchronisation des deux images

Maintenant, nous allons analyser la situation engendrée par les deux webcams. Nous avons vu, dans le chapitre 5, que nous synchronisions les deux images à l'aide d'un signal lumineux. Nous allons constater que ce signal lumineux n'est pas suffisant pour synchroniser correctement les deux images. Considérons, tout d'abord, le premier cas où le signal de synchronisation est perçu par les webcams après le même laps de temps depuis le début de leur intégration. Nous faisons l'hypothèse que le temps d'intégration t_i et le délai de transfert δ sont identiques pour les deux caméras ; dans ce cas, nous nous trouvons dans la situation représentée par la figure 8.6. Considérons la fonction

$$Image_j^W = \mathcal{I}_{NT}(t_{initial},d,W).$$

Cette fonction $\mathcal{I}_{NT}$ intègre l'image $Image_j^W$ à partir d'un instant initial $t_{initial}$ pendant une certaine durée d pour une webcam W. Le numéro de l'image est représenté par j. L'image $Image_j^{W_1} = \mathcal{I}_{NT}(t_1,d_1,W_1)$ correspond à l'$Image_k^{W_2} = \mathcal{I}_{NT}(t_2,d_2,W_2)$ si et

FIG. 8.5 – *Traînée de la balle dans l'image provenant d'une webcam.*

seulement si $t_1 = t_2$ et $d_1 = d_2$. Dans ce cas, on note

$$Image_j^{W_1} \leftrightarrow Image_k^{W_2}.$$

Dans le cas contraire nous notons que

$$Image_j^{W_1} \nleftrightarrow Image_k^{W_2},$$

ce qui signifie que les webcams W_1 et W_2 sont désynchronisées. Dans notre première situation (figure 8.6), nous avons

$$Image_1^{Webcam_1} = \mathcal{I}_{NT}(t_0 + kh - t_i - \delta, t_i, Webcam_1),$$

$$Image_1^{Webcam_2} = \mathcal{I}_{NT}(t_0 + kh - t_i - \delta, t_i, Webcam_2).$$

Nous pouvons donc dire que

$$Image_1^{Webcam_1} \leftrightarrow Image_1^{Webcam_2}.$$

Dès lors, les deux webcams sont parfaitement synchronisées. Cette situation est idéale et les données extraites sont relatives à la même portion de la trajectoire. Cependant, il ne faut pas oublier la présence de bruit de pixélisation et de traitement d'images.

A-t-on la garantie d'être dans cette situation? Non.

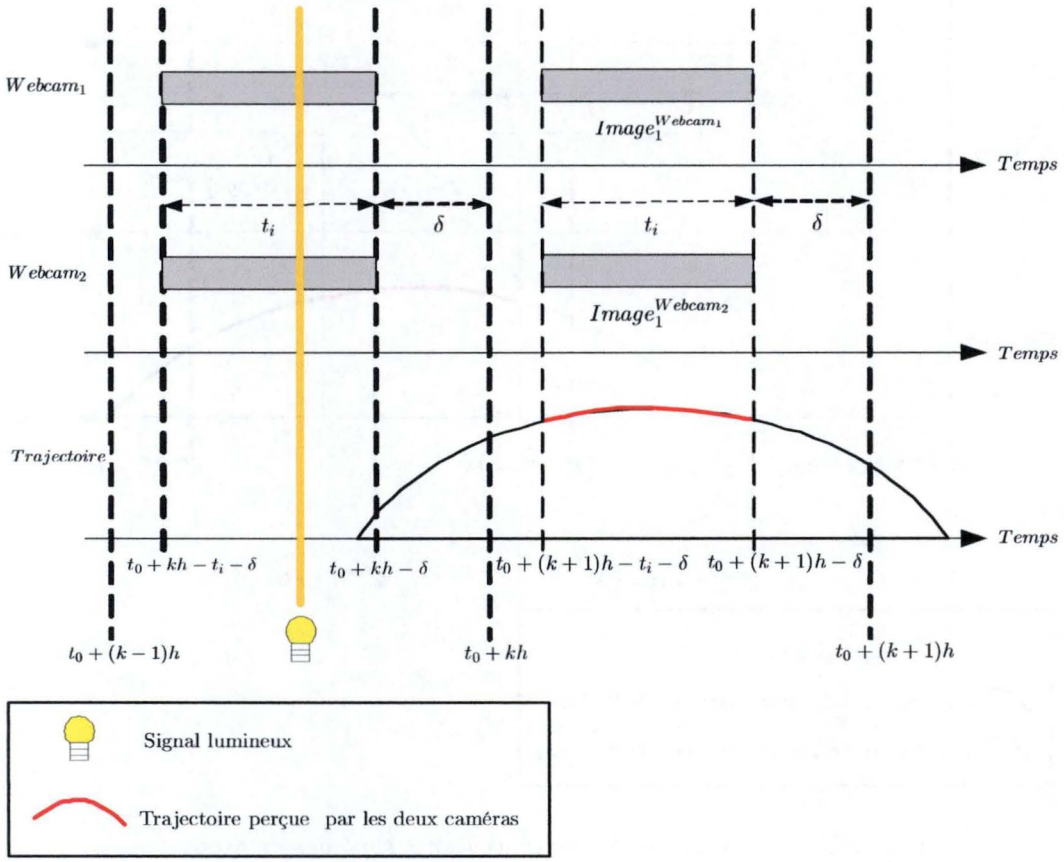


FIG. 8.6 – *Situation où les deux webcams sont parfaitement synchronisées.*

Pour démontrer cela, nous allons considérer le cas le plus extrême possible avec ce système de synchronisation. Le signal de synchronisation peut être perçu par une des caméras au début d'une période d'intégration et par l'autre à la fin d'une période d'intégration. Dans ce cas, les deux traînées de la balle extraites de l'image sur lesquelles on va calculer les centres de gravité ne correspondent plus à la même partie de la trajectoire. En effet, nous avons

$$Image_1^{Webcam_1} = \mathcal{I}_{NT}(t_0^{Webcam_1} + kh - t_i - \delta, t_i, Webcam_1),$$

$$Image_1^{Webcam_2} = \mathcal{I}_{NT}(t_0^{Webcam_2} + kh - t_i - \delta, t_i, Webcam_2)$$

où

$$t_0^{Webcam_1} \neq t_0^{Webcam_2}.$$

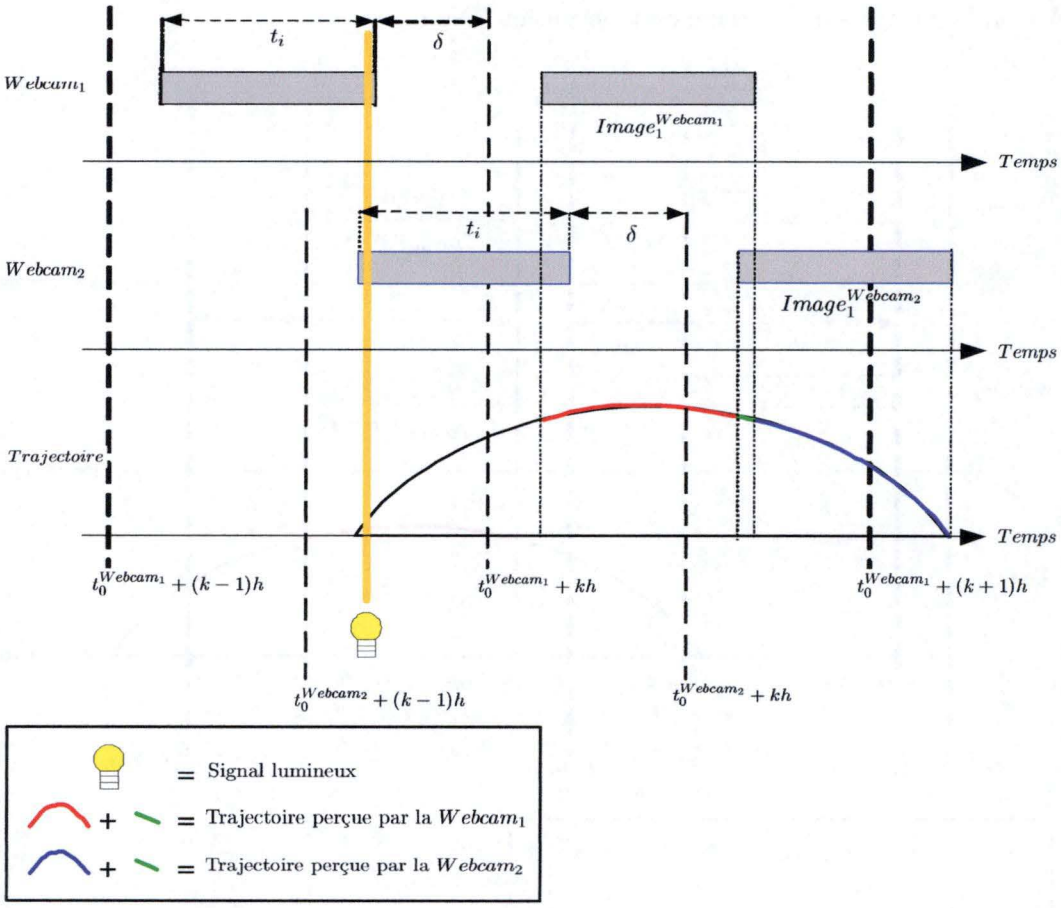


FIG. 8.7 – *Situation où les deux webcams sont désynchronisées.*

Nous pouvons donc dire que

$$Image_1^{Webcam_1} \nleftrightarrow Image_1^{Webcam_2}.$$

Dès lors, les webcams ne sont plus synchronisées. Cette situation est illustrée à la figure 8.7. Les centres de gravité extraits des deux images ne correspondent pas au même emplacement de la balle dans le temps.

En pratique, nous allons avoir des situations qui se trouvent entre les deux cas extrêmes cités ci-dessus. Nous avons tenter de résoudre le problème de synchronisation en utilisant un stroboscope. Cette solution s'est avérée inapplicable par manque de contrôle des webcams.

8.4 Conséquences

Deux conséquences découlent de ces constatations. La première concerne la section 8.2. Dans un cas d'acquisition parfaite, un bruit léger de quelques pixels sera présent sur la mesure du centre de gravité de la balle. Des techniques telles que la régression polynômiale ou, éventuellement, des techniques utilisant des régressions linéaires peuvent donner un meilleur résultat pour prédire à long terme. Cependant, dans un cas pareil, le processus de régression ne sera valable que pour une trajectoire et il faudra le recommencer pour chaque trajectoire observée.

La deuxième conséquence est issue de la section 8.3. Les caméras doivent être considérées de manière séparée dans la constitution d'un modèle. En effet, nous avons vu que l'algorithme ne garantissait pas la synchronisation des données des deux caméras et pouvait engendrer une désynchronisation assez forte qui est fonction de la période d'intégration.

8.5 Conclusion

Nous avons vu que, suite à la pixélisation et au matériel utilisé, en l'occurrence des webcams, nous allions devoir changer de vision pour résoudre le problème de prédiction de trajectoire à long terme en vue, par exemple, d'une interception. Nous allons donc opter pour un point de vue fort différent par rapport aux chapitres précédents.

Pour cela, nous allons étudier dans les chapitres suivants, des méthodes qui considèrent les caméras de manière séparée et qui utilisent des modèles plus robustes au bruit en appliquant des techniques de régression. Leur performance sera donc évaluée en fonction du nombre d'observations nécessaires pour prédire à long terme dans un voisinage acceptable du point d'interception.

Chapitre 9

Modèle polynômial de prédiction dans l'image

9.1 Introduction

Nous avons vu au chapitre précédent que, réaliser un modèle qui prendrait comme entrée les positions de la balle dans l'image, pose de gros problèmes à cause du bruit induit par la pixélisation et la désynchronisation des webcams.

Dans ce chapitre, nous choisissons une nouvelle approche. Les trajectoires sont de type parabolique et leurs coordonnées dans les images évoluent aussi suivant des paraboles. Nous allons donc essayer de prédire en utilisant de simples régressions polynômiales dont l'entrée représente le temps, une donnée non bruitée. Ce chapitre n'exploite aucune connaissance du modèle sous-jacent. De plus, au chapitre 8, nous avons constaté qu'il fallait considérer les images séparément car elles étaient désynchronisées. Nous allons donc prédire en tenant compte uniquement des observations d'une caméra. Notre méthode sera applicable, aussi, à la seconde.

9.2 Principes du modèle

Les principes du modèle sont très simples. Nous n'allons pas considérer les caméras ensemble mais séparément. De plus, nous allons directement traiter le problème dans l'image. La figure 9.1, représente la variation des coordonnées d'une trajectoire réelle dans l'image en fonction du temps. Si nous envisageons les deux coordonnées séparément dans une image en fonction du temps, nous voyons que ces coordonnées ont une forme parabolique. Dès lors, des polynômes dont le terme variable est le temps, peuvent très bien nous donner une approximation de ces coordonnées au cours de la trajectoire.

Nous proposons donc d'effectuer une régression polynômiale pour chacune des coordonnées. Les degrés des polynômes seront déterminés empiriquement. Nous retiendrons ceux qui donnent les meilleurs résultats. Nous ajusterons les régressions de ces polynômes en fonction des points observés de la trajectoire. Ainsi, au début, nous effectuerons une régression sur, par exemple, 5 points, puis 6, ...

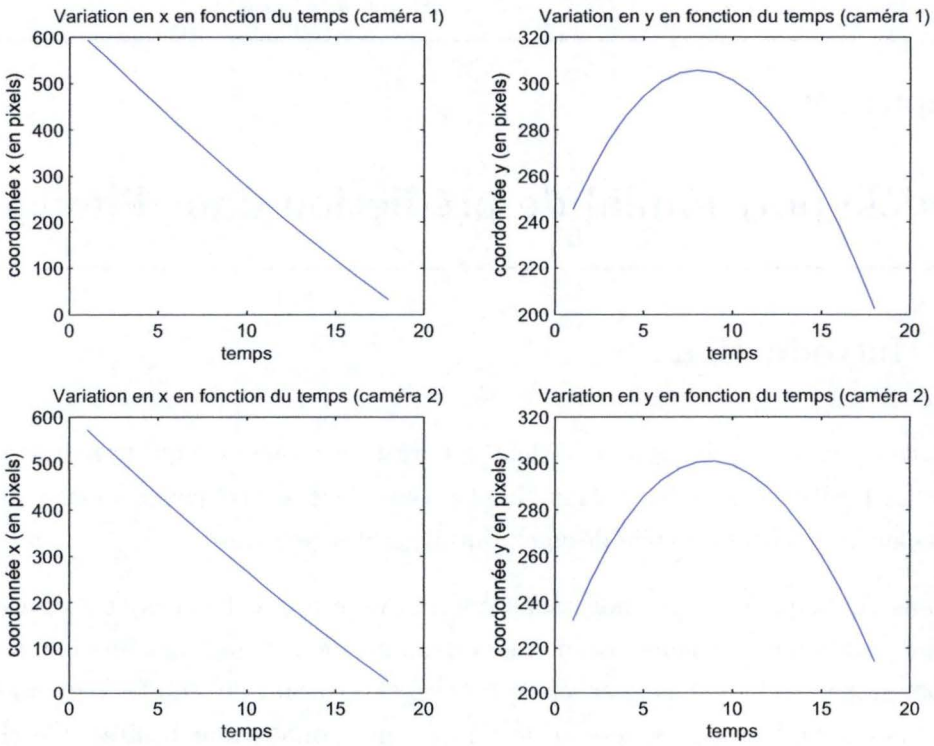


FIG. 9.1 – *Variation des coordonnées x et y d'une trajectoire réelle.*

9.3 Régression polynômiale

Nous allons présenter le principe de régression polynômiale pour un polynôme de degré quelconque. Soient y_i les données observées au cours du temps. Considérons le polynôme de degré k qui doit réaliser au mieux l'approximation de ces données.

$$y = a_k x^k + \cdots + a_2 x^2 + a_1 x + a_0. \tag{9.1}$$

Nous cherchons à minimiser l'erreur au carré entre les observations réelles et le modèle polynômial. Considérons que nous avons n observations. Nous cherchons donc à minimiser la fonction

$$E^2 = \sum_{i=1}^{n} (y_i - (a_k x_i^k + \cdots + a_2 x_i^2 + a_1 x_i + a_0))^2. \tag{9.2}$$

Pour atteindre un minimum, nous devons satisfaire les contraintes

$$\begin{aligned}
\frac{\delta(E^2)}{\delta a_0} &= -2\sum_{i=1}^{n}(y_i - (a_k x_i^k + \cdots + a_2 x_i^2 + a_1 x_i + a_0)) &= 0,\\
\frac{\delta(E^2)}{\delta a_1} &= -2\sum_{i=1}^{n}(y_i - (a_k x_i^k + \cdots + a_2 x_i^2 + a_1 x_i + a_0))x &= 0,\\
&\vdots\\
\frac{\delta(E^2)}{\delta a_k} &= -2\sum_{i=1}^{n}(y_i - (a_k x_i^k + \cdots + a_2 x_i^2 + a_1 x_i + a_0))x^k &= 0.
\end{aligned}$$

Dès lors, nous pouvons tirer les équations

$$\begin{aligned}
a_o n + a_1 \sum_{i=1}^{n} x_i + \cdots + a_k \sum_{i=1}^{n} x_i^k &= \sum_{i=1}^{n} y_i,\\
a_o \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + \cdots + a_k \sum_{i=1}^{n} x_i^{k+1} &= \sum_{i=1}^{n} x_i y_i,\\
&\vdots\\
a_o \sum_{i=1}^{n} x_i^k + a_1 \sum_{i=1}^{n} x_i^{k+1} + \cdots + a_k \sum_{i=1}^{n} x_i^{2k} &= \sum_{i=1}^{n} x_i^k y_i.
\end{aligned}$$

Cela peut s'exprimer en notation matricielle par

$$\left(\sum_{i=1}^{n}\begin{pmatrix} x_i^{2k} & \cdots & x_i^{k+1} & x_i^k \\ \vdots & \ddots & \vdots & \vdots \\ x_i^{k+1} & \cdots & x_i^2 & x_i \\ x_i^k & \cdots & x_i & 1 \end{pmatrix}\right)\begin{pmatrix} a_k \\ \vdots \\ a_1 \\ a_0 \end{pmatrix} = \sum_{i=1}^{n}\begin{pmatrix} x_i^k y_i \\ \vdots \\ x_i y_i \\ y_i \end{pmatrix}.$$

Appelons

$$C = \left(\sum_{i=1}^{n}\begin{pmatrix} x_i^{2k} & \cdots & x_i^{k+1} & x_i^k \\ \vdots & \ddots & \vdots & \vdots \\ x_i^{k+1} & \cdots & x_i^2 & x_i \\ x_i^k & \cdots & x_i & 1 \end{pmatrix}\right),$$

$$B = \sum_{i=1}^{n}\begin{pmatrix} x_i^k y_i \\ \vdots \\ x_i y_i \\ y_i \end{pmatrix}.$$

Notre système devient

$$C \begin{pmatrix} a_k \\ \vdots \\ a_1 \\ a_0 \end{pmatrix} = B.$$

La solution du système est donnée par l'équation

$$\begin{pmatrix} a_k \\ \vdots \\ a_1 \\ a_0 \end{pmatrix} = C^{-1}B. \tag{9.3}$$

Considérons la fonction

$$\mathcal{P}_{OLY} = \mathbf{REGR}(d,\mathcal{A})$$

qui réalise la régression polynômiale de degré d pour un ensemble $\mathcal{A}$ constitué des couples (y_i,x_i) et qui renvoie le polynôme ainsi obtenu $\mathcal{P}_{OLY}$.
$\mathcal{P}_{OLY}$ prend une valeur x et renvoie le résultat y correspondant

$$y = \mathcal{P}_{OLY}(x).$$

9.4 Méthode

Considérons une trajectoire observée par une des deux caméras, qui a débuté à l'instant t_0 et se termine à l'instant $t_0 + t_f h$. La méthode utilisée est illustrée à la figure 9.2. A l'instant $t_0 + kh$, $k \in \mathbb{N}$, $k \geq d$ et $k \leq t_f$, nous possédons deux ensembles d'observations, un ensemble par coordonnée. Ces ensembles valent

$$\mathcal{O}_x(k) = \{x(t_0 + jh) \mid 0 \leq j \leq k\},$$

$$\mathcal{O}_y(k) = \{y(t_0 + jh) \mid 0 \leq j \leq k\}.$$

Pour chacun de ces ensembles, nous pouvons constituer des couples, observation-temps, pour effectuer une régression polynômiale. Nous obtenons comme ensembles de régression

$$\mathcal{A}_x(k) = \{(x(t_0 + jh),j) \mid 0 \leq j \leq k\},$$

$$\mathcal{A}_y(k) = \{(y(t_0 + jh),j) \mid 0 \leq j \leq k\}.$$

Nous pouvons maintenant effectuer une régression polynômiale pour chacune des coordonnées.

$$\mathcal{P}^k_{OLY_x} = \mathbf{REGR}(d,\mathcal{A}_x(k)),$$

$$\mathcal{P}^k_{OLY_y} = \mathbf{REGR}(d,\mathcal{A}_y(k)).$$

Nous pouvons maintenant prédire un point d'interception dans l'image $p = (x,y)$ à l'instant $t \geq k$ à l'aide de ces polynômes.

$$p = \left(\mathcal{P}^k_{OLY_x}(t),\mathcal{P}^k_{OLY_y}(t)\right).$$

A l'instant suivant $t_0 + (k+1)h$, nous recommençons le processus avec $k := k+1$ afin d'obtenir des polynômes qui nous donneront une prédiction plus précise. Remarquons que cette méthode ne réalise aucun apprentissage. En effet, chacun des polynômes trouvés n'est valable que pour la trajectoire considérée.

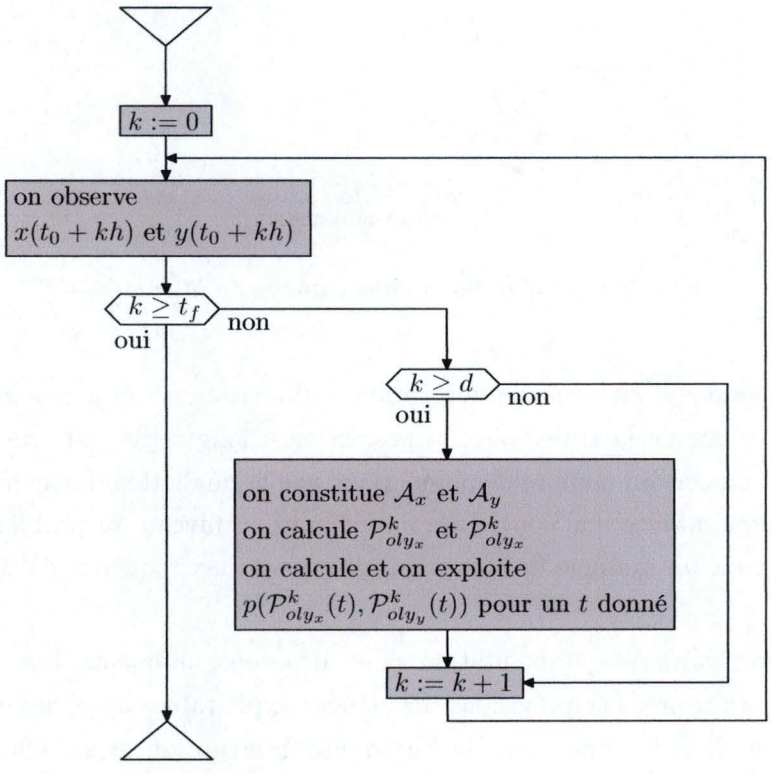


FIG. 9.2 – *Méthode de prédiction polynômiale.*

9.5 Résultats des simulations

Nos simulations ont été réalisées sur des données réelles issues de deux webcams. Ces webcams nous procurent une trentaine d'images par seconde. A cause de nos conditions d'expérimentation, les trajectoires observées durent moins d'une seconde.

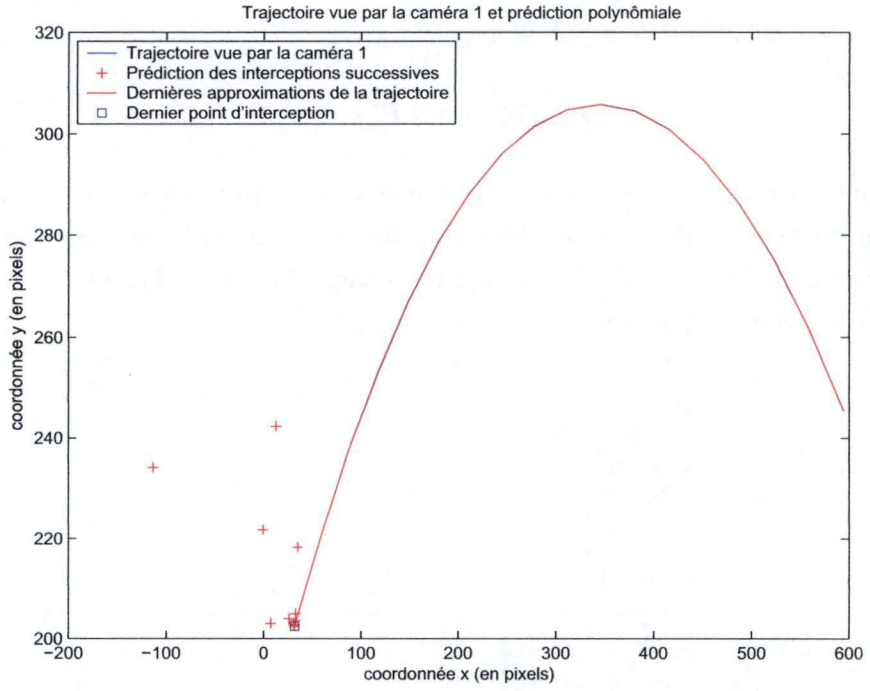


FIG. 9.3 – *Prédictions dans l'image de la caméra 1.*

Les prédictions d'un point d'interception s'affinent au fur et à mesure (figures 9.3 et 9.4). Au début de la trajectoire, la prédiction à long terme est très mauvaise. Il faut, en fait, un certain nombre de points pour que la prédiction devienne exploitable. Cette technique ajoute une contrainte importante au niveau du problème. Il faudra donc opter pour un système de vision qui dispose d'une fréquence d'échantillonnage très élevée.

Dans notre cadre d'expérimentation, nous disposons, au mieux, d'une vingtaine de points par trajectoire. Les prévisions deviennent exploitables après un certain temps d'observation. Il faut donc, dans le but d'une interception avec l'effecteur, que le mouvement puisse être assez rapide.

Au niveau de l'erreur de prédiction des derniers polynômes (figure 9.5), nous voyons qu'avec des polynômes de degré 4, nous obtenons une erreur inférieure au pixel. Cette erreur est satisfaisante pour l'objectif fixé. Cependant, cette approche ne procure aucun apprentissage ni aucune mémorisation. Nous devons recommencer le procédé

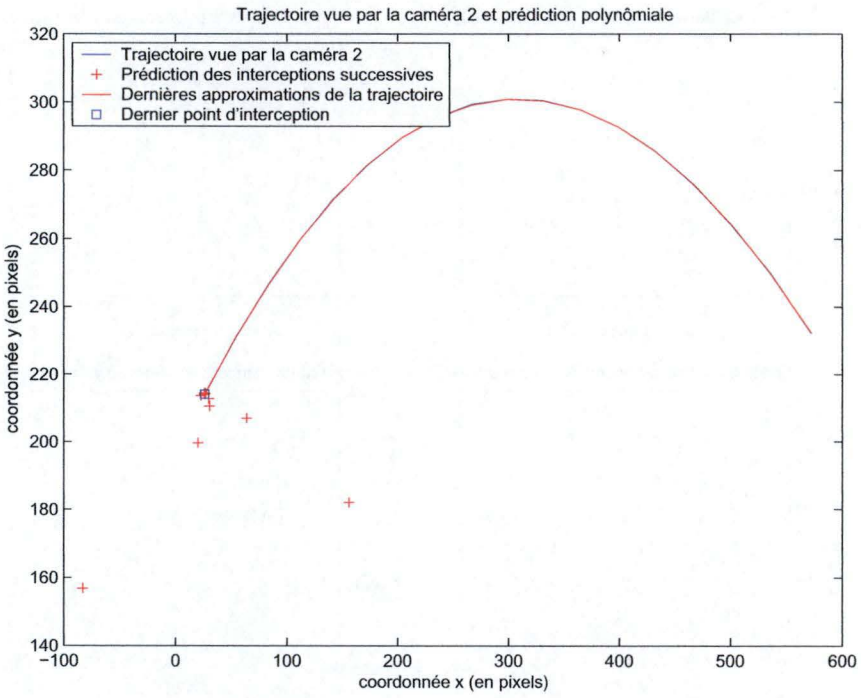


FIG. 9.4 – *Prédictions dans l'image de la caméra 2.*

de régression à chaque trajectoire. Dès lors, cette erreur est significative de l'erreur du dernier polynôme mais non de l'erreur au cours de la trajectoire.

9.6 Mise en oeuvre

Pour rendre cette approche exploitable, il faut se fixer un horizon de prédiction qui se trouve dans l'image limitée par sa résolution. L'idéal serait de pouvoir choisir un point d'interception qui se situerait dans une position adéquate par rapport à la position de l'effecteur et à la trajectoire considérée pour laisser le temps au bras robot de se déplacer. Ainsi, dès le début, on déplacerait l'effecteur du bras à une position bien située et on ajusterait progressivement la position.

Les conditions de lancer vont donc directement influencer la réussite de l'interception. En effet, l'espace accessible par l'effecteur est réduit. De plus, l'espace visible par le système de caméras stéréoscopiques est lui aussi réduit. Il faut, donc, essayer de tirer un maximum de profit des premières observations de la trajectoire.

Enfin, remarquons qu'il serait possible de bénéficier plus rapidement de prévisions précises si on arrivait à effectuer un certain apprentissage. En effet, selon la méthode présentée dans ce chapitre, si deux trajectoires identiques sont générées, on doit recommencer le processus de régression et d'ajustement, alors que, si un certain

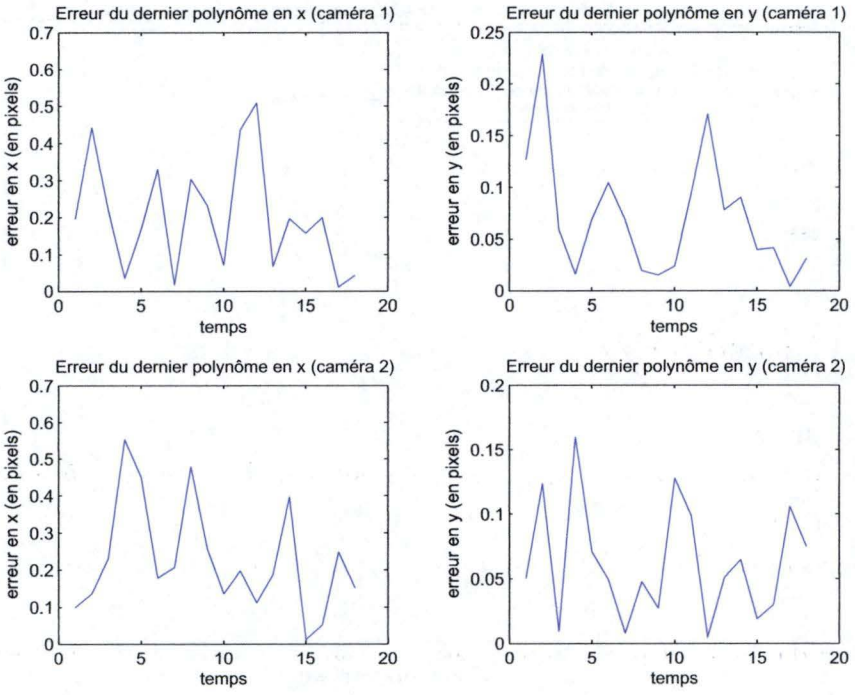


FIG. 9.5 – *Erreur de prédiction du dernier polynôme.*

apprentissage avait été réalisé, on pourrait directement bénéficier des prévisions après ajustement.

9.7 Conclusion

Cette approche est assez simple et elle solutionne le problème de prédiction. Cependant, elle fournit tardivement des prédictions à long terme exploitables. En effet, il faut avoir parcouru une grande partie de la trajectoire avant de pouvoir obtenir une bonne prédiction.

Dans le chapitre suivant, nous allons donc continuer à utiliser les polynômes, mais nous tenterons de trouver une méthode plus efficace afin d'en tirer profit. Nous continuerons, comme tout au long de ce document, à ignorer les paramètres de la scène. Ceux-ci doivent, soit être appris, soit ne pas être nécessaires au modèle.

Chapitre 10

Modèle multi-polynômial de prédiction dans l'image

10.1 Introduction

Ce chapitre tente d'exploiter les enseignements des modèles antérieurs. Dans le chapitre précédent, nous avons remarqué que, pour la prédiction à long terme, un modèle faisant intervenir le temps est plus robuste au bruit. Ce chapitre va explorer une nouvelle voie. Nous allons voir comment créer un modèle qui aurait comme entrée le temps en considérant le modèle sous-jacent. Nous mesurerons ensuite son efficacité en terme d'erreur de prédiction. Nous déterminerons aussi si ce modèle peut être appliqué en pratique, c'est-à-dire s'il nécessite peu d'observations pour prédire un point d'interception correct.

10.2 Hypothèses

Nous ne considérons pas la force de frottement dans les développements théoriques. Cependant, celle-ci sera présente dans les simulations. En effet, les simulations seront réalisées sur des données de trajectoires réelles.

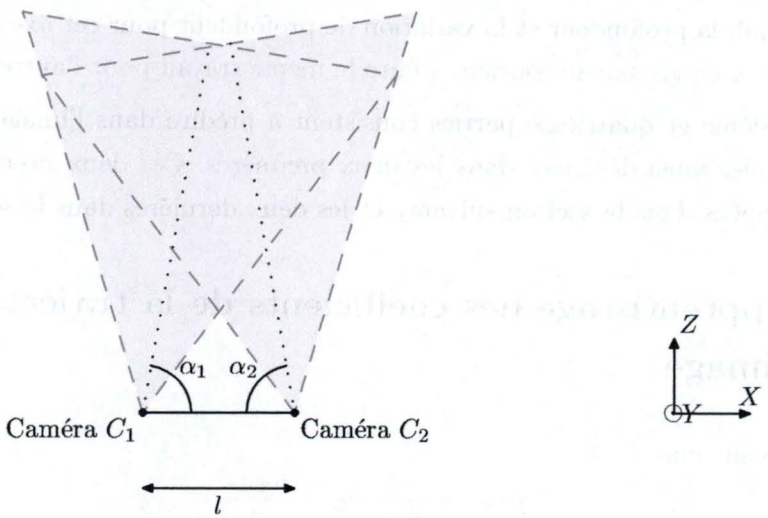


FIG. 10.1 – *Vue du dessus de la scène.*

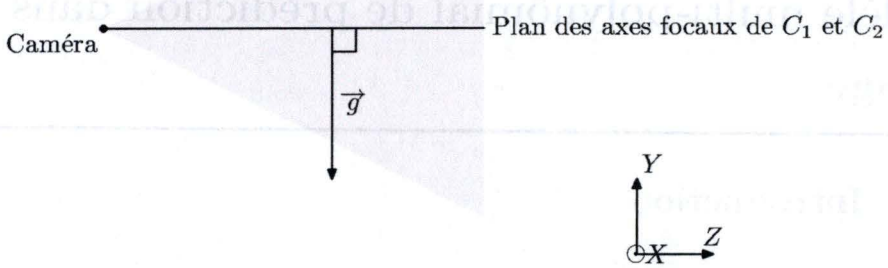


FIG. 10.2 – *Vue latérale de la scène.*

Les caméras sont dans une configuration quelconque. Elles sont seulement placées dans un même plan. Par commodité de développement, nous considérons que le plan des axes optiques des caméras est perpendiculaire à la force de pesanteur. La scène est représentée par les figures 10.1 et 10.2.

10.3 Méthode

La méthode proposée est illustrée à la figure 10.3. Cette méthode itérative se décompose en quatre parties. A chaque instant t, il faudra recommencer les quatre étapes afin d'affiner les prédictions. La première étape consiste à retrouver les caractéristiques de la trajectoire pour une coordonnée dans l'image. Ainsi, on déterminera la vitesse, le point initial, la profondeur et la variation de profondeur pour cet axe dans l'espace cartésien. La seconde partie consiste à faire le même travail pour l'autre coordonnée.

Les troisième et quatrième parties consistent à prédire dans l'image à l'aide des caractéristiques ainsi dégagées dans les deux premières. Ces deux premières parties sont développées dans la section suivante et les deux dernières dans la section 10.6.

10.4 Apprentissage des coefficients de la trajectoire dans l'image

Nous savons que

$$p_{Im_x} = \frac{f}{\Psi_x p_z} p_x + \frac{u_{max}}{2}$$

et

$$p_{Im_y} = \frac{f}{\Psi_y p_z} p_y - \frac{v_{max}}{2}.$$

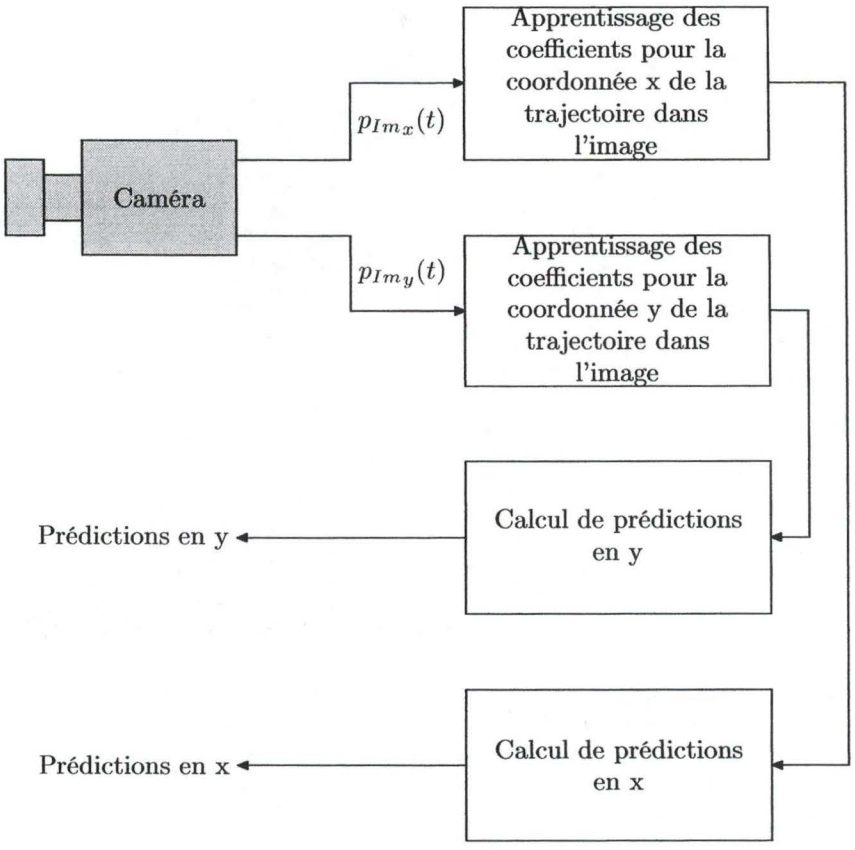


FIG. 10.3 – *Méthode de prédiction.*

Nous pouvons alors écrire

$$p_{Im_x} - \frac{u_{max}}{2} = \frac{f}{\Psi_x p_z} p_x$$

et

$$p_{Im_y} + \frac{v_{max}}{2} = \frac{f}{\Psi_y p_z} p_y.$$

Dès lors, nous avons

$$p^{new}_{Im_x}(t) = \omega_1 \frac{p_x}{p_z}, \tag{10.1}$$

$$p^{new}_{Im_y}(t) = \omega_2 \frac{p_y}{p_z} \tag{10.2}$$

avec

$$p^{new}_{Im_x}(t) = p_{Im_x} - \frac{u_{max}}{2},$$

$$p^{new}_{Im_y}(t) = p_{Im_y} + \frac{v_{max}}{2},$$

$$\omega_1 = \frac{f}{\Psi_x},$$

$$\omega_2 = \frac{f}{\Psi_y}.$$

Nous pouvons considérer ω_1 et ω_2 comme des facteurs d'échelle. En effet, ces deux valeurs sont fixes pour chaque caméra.

D'autre part, nous savons que

$$p_x(t) = p_x(0) + v_x(0)t,$$

$$p_y(t) = p_y(0) + v_y(0)t + \frac{g}{2}t^2,$$

$$p_z(t) = p_z(0) + v_z(0)t.$$

En utilisant ces équations, nous pouvons reformuler les équations 10.1 et 10.2

$$p^{new}_{Im_x}(t) = \omega_1 \frac{p_x(0) + v_x(0)t}{p_z(0) + v_z(0)t},$$

$$p^{new}_{Im_y}(t) = \omega_2 \frac{p_y(0) + v_y(0)t + \frac{g}{2}t^2}{p_z(0) + v_z(0)t}.$$

Distribuons les termes ω_1 et ω_2,

$$p^{new}_{Im_x}(t) = \frac{\omega_1 p_x(0) + \omega_1 v_x(0)t}{p_z(0) + v_z(0)t}, \tag{10.3}$$

$$p^{new}_{Im_y}(t) = \frac{\omega_2 p_y(0) + \omega_2 v_y(0)t + \omega_2 \frac{g}{2}t^2}{p_z(0) + v_z(0)t}. \tag{10.4}$$

En multipliant le membre de gauche de ces équations par le dénominateur du membre de droite correspondant, nous avons

$$p^{new}_{Im_x}(t)p_z(0) + p^{new}_{Im_x}(t)v_z(0)t = \omega_1 p_x(0) + \omega_1 v_x(0)t,$$

$$p^{new}_{Im_y}(t)p_z(0) + p^{new}_{Im_y}(t)v_z(0)t = \omega_2 p_y(0) + \omega_2 v_y(0)t + \omega_2 \frac{g}{2}t^2.$$

Exprimons ces équations sous la forme matricielle :

$$\begin{pmatrix} p^{new}_{Im_x}(t) \\ p^{new}_{Im_x}(t)t \\ 1 \\ t \end{pmatrix}^T \begin{pmatrix} p_z(0) \\ v_z(0) \\ -\omega_1 p_x(0) \\ -\omega_1 v_x(0) \end{pmatrix} = 0, \tag{10.5}$$

$$\begin{pmatrix} p^{new}_{Im_y}(t) \\ p^{new}_{Im_y}(t)t \\ 1 \\ t \end{pmatrix}^T \begin{pmatrix} \frac{p_z(0)}{\omega_2} \\ \frac{v_z(0)}{\omega_2} \\ -p_y(0) \\ -v_y(0) \end{pmatrix} = \frac{g}{2}t^2.$$

De manière équivalente, nous pouvons écrire, à partir de l'équation 10.5,

$$\begin{pmatrix} p^{new}_{Im_x}(t) \\ p^{new}_{Im_x}(t)t \\ t \end{pmatrix}^T \begin{pmatrix} \frac{p_z(0)}{\omega_1 p_x(0)} \\ \frac{v_z(0)}{\omega_1 p_x(0)} \\ -\frac{v_x(0)}{p_x(0)} \end{pmatrix} = 1.$$

Nous déterminerons par régression, les coefficients des deux matrices. A cette fin, nous utiliserons les premiers points de la trajectoire et nous calculerons la prédiction de la trajectoire. A chaque nouveau point, nous préciserons ces coefficients. Considérons t_o l'instant initial de la trajectoire et k le $k^{ième}$ point observé. Par facilité, nous posons que $t_0 = 0$. Nous avons comme ensembles d'apprentissage

$$\mathcal{A}_x(k) = \{(1, IN_x(j)) \mid 0 \leq j \leq k\},$$
$$\mathcal{A}_y(k) = \left\{\left(\frac{g}{2}j^2, IN_y(j)\right) \mid 0 \leq j \leq k\right\}$$

avec

$$IN_x(k) = \begin{pmatrix} p^{new}_{Im_x}(k) \\ p^{new}_{Im_x}(k)k \\ k \end{pmatrix},$$

$$IN_y(k) = \begin{pmatrix} p^{new}_{Im_y}(k) \\ p^{new}_{Im_y}(k)k \\ 1 \\ k \end{pmatrix}.$$

Nous pouvons maintenant effectuer une régression linéaire pour chacune des coordonnées.

$$C^k_{oefficients_x} = \mathbf{REGRL}(\mathcal{A}_x(k)) = \begin{pmatrix} c^k_{1_x} \\ c^k_{2_x} \\ c^k_{3_x} \end{pmatrix},$$

$$C^k_{oefficients_y} = \mathbf{REGRL}(\mathcal{A}_y(k)) = \begin{pmatrix} c^k_{1_y} \\ c^k_{2_y} \\ c^k_{3_y} \\ c^k_{4_y} \end{pmatrix}.$$

Remarquons que nous avons admis que nous connaissions la gravité. Nous aurions pu remplacer ce terme par une constante. Dans ce cas, un deuxième facteur d'échelle serait intervenu. Mais les résultats sont identiques.

10.5 Régression linéaire multiple

Nous allons présenter le principe de régression linéaire multiple pour un nombre quelconque de coefficients. Soient y_i les données observées au cours du temps. Considérons

$$y = a_k x_k + \cdots + a_2 x_2 + a_1 x_1. \tag{10.6}$$

Nous cherchons à minimiser l'erreur au carré entre les observations réelles et le modèle linéaire. Considérons que nous avons n observations. Nous cherchons donc à minimiser la fonction

$$E^2 = \sum_{i=1}^{n} (y_i - (a_k x_{k_i} + \cdots + a_2 x_{2_i} + a_1 x_{1_i}))^2. \tag{10.7}$$

Pour atteindre un minimum, nous devons satisfaire les contraintes

$$\begin{aligned} \frac{\delta(E^2)}{\delta a_1} &= -2 \textstyle\sum_{i=1}^{n} (y_i - (a_k x_{k_i} + \cdots + a_2 x_{2_i} + a_1 x_{1_i})) x_1 &= 0, \\ &\vdots \\ \frac{\delta(E^2)}{\delta a_k} &= -2 \textstyle\sum_{i=1}^{n} (y_i - (a_k x_{k_i} + \cdots + a_2 x_{2_i} + a_1 x_{1_i})) x_k &= 0. \end{aligned}$$

Dès lors, nous pouvons tirer les équations

$$\begin{aligned} a_1 \textstyle\sum_{i=1}^{n} x_{1_i}^2 + \cdots + a_k \sum_{i=1}^{n} x_{1_i} x_{k_i} &= \textstyle\sum_{i=1}^{n} x_{1_i} y_i, \\ &\vdots \\ a_1 \textstyle\sum_{i=1}^{n} x_{k_i} x_{1_i} + \cdots + a_k \sum_{i=1}^{n} x_{k_i}^2 &= \textstyle\sum_{i=1}^{n} x_{k_i} y_i. \end{aligned}$$

Cela peut s'exprimer en notation matricielle par

$$\left(\sum_{i=1}^{n} \begin{pmatrix} x_{k_i}^2 & \cdots & x_{k_i} x_{1_i} \\ \vdots & \ddots & \vdots \\ x_{k_i} x_{1_i} & \cdots & x_{1_i}^2 \end{pmatrix} \right) \begin{pmatrix} a_k \\ \vdots \\ a_1 \end{pmatrix} = \sum_{i=1}^{n} \begin{pmatrix} x_{k_i} y_i \\ \vdots \\ x_{1_i} y_i \end{pmatrix}.$$

Appelons

$$C = \left(\sum_{i=1}^{n} \begin{pmatrix} x_{k_i}^2 & \dots & x_{k_i} x_{1_i} \\ \vdots & \ddots & \vdots \\ x_{k_i} x_{1_i} & \dots & x_{1_i}^2 \end{pmatrix} \right),$$

$$B = \sum_{i=1}^{n} \begin{pmatrix} x_{k_i} y_i \\ \vdots \\ x_{1_i} y_i \end{pmatrix}.$$

Notre système devient

$$C \begin{pmatrix} a_k \\ \vdots \\ a_1 \end{pmatrix} = B,$$

alors, la solution du système est donnée par l'équation

$$\begin{pmatrix} a_k \\ \vdots \\ a_1 \end{pmatrix} = C^{-1} B. \tag{10.8}$$

Considérons la fonction

$$Coeff = \mathbf{REGRLM}(\mathcal{A})$$

qui réalise la régression linéaire multiple pour un ensemble $\mathcal{A}$ constitué des couples $(y_i, (x_{1_i}, x_{2_i}, \dots, x_{k_i})^T)$ et qui renvoie les coefficients ainsi obtenus

$$Coeff = \begin{pmatrix} a_k \\ \vdots \\ a_1 \end{pmatrix}.$$

10.6 Calcul de prédictions

A partir des coefficients déterminés dans la section précédente, nous allons utiliser les équations 10.3 et 10.4 pour prédire le reste de la trajectoire.

$$p_{Im_x}^{new}(k) = p_{Im_x}^{new}(t) = \frac{\omega_1 p_x(0) + \omega_1 v_x(0) t}{p_z(0) + v_z(0) t} \approx \frac{1 - c_{3_x}^k k}{c_{1_x}^k + k c_{2_x}^k},$$

$$p_{Im_y}^{new}(k) = p_{Im_y}^{new}(t) = \frac{\omega_2 p_y(0) + \omega_2 v_y(0)t + \omega_2 \frac{g}{2}t^2}{p_z(0) + v_z(0)t} \approx \frac{-c_{3_y}^k - kc_{4_y}^k + \frac{g}{2}k^2}{c_{1_y}^k + c_{2_y}^k k}.$$

Nous pouvons prédire à n'importe quel horizon souhaité. En effet, nous utilisons comme seule entrée, le temps dont l'échelle est établie par convention. Remarquons que le calcul à partir des coefficients est une approximation du point. Cette approximation se précise au cours de la trajectoire.

10.7 Résultats de simulations sur des données réelles

Les résultats sont présentés aux figures 10.4, 10.5 et 10.6. Nous voyons, dans les images caméras, que la prédiction à long terme est déjà exploitable dès les cinq premiers points de la trajectoire. Lors de l'avancement de la trajectoire, nous allons pouvoir affiner cette prédiction pour, au final, atteindre une grande précision.

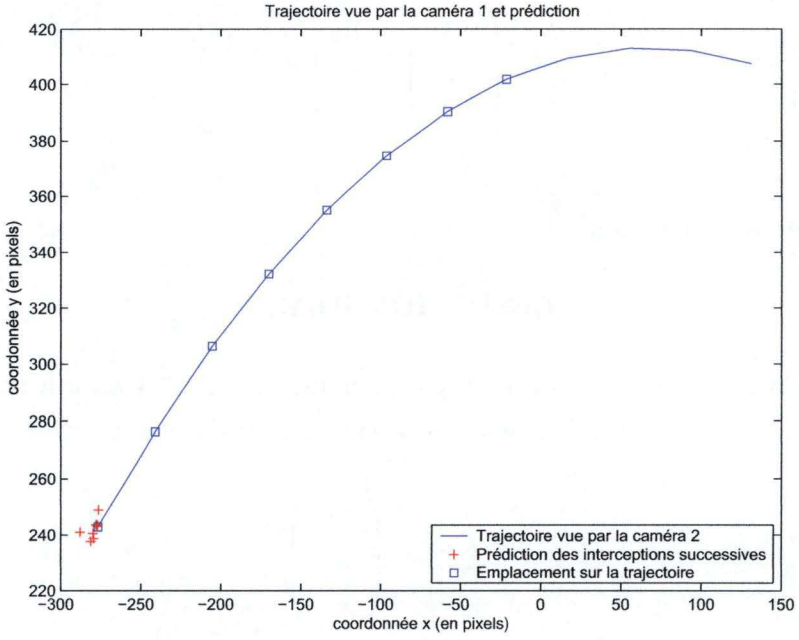


FIG. 10.4 – *Prédictions pour la caméra 1.*

L'erreur de prédiction du point d'interception diminue très rapidement (figure 10.6). Elle peut commencer aux alentours de 10 pixels et très rapidement atteindre une valeur en dessous du pixel : résultats corrects pour une application pratique.

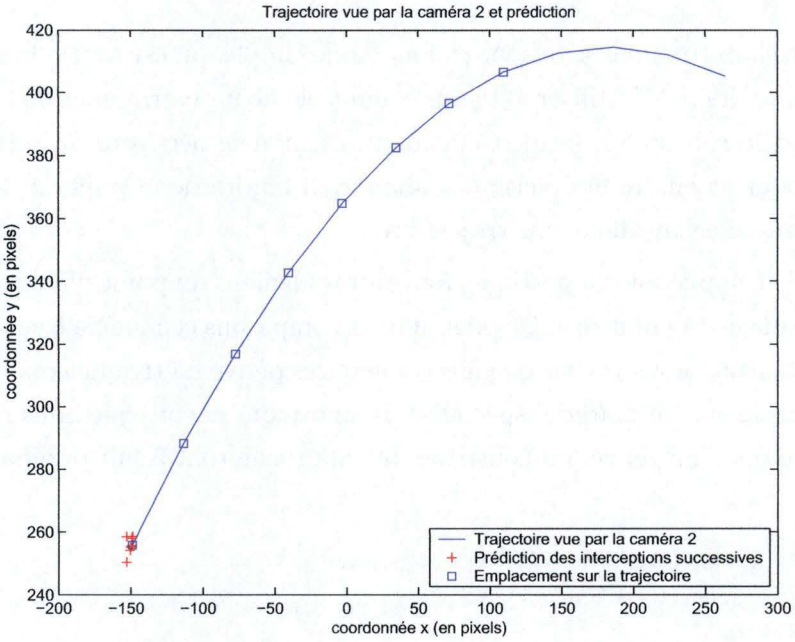


FIG. 10.5 – *Prédictions pour la caméra 2.*

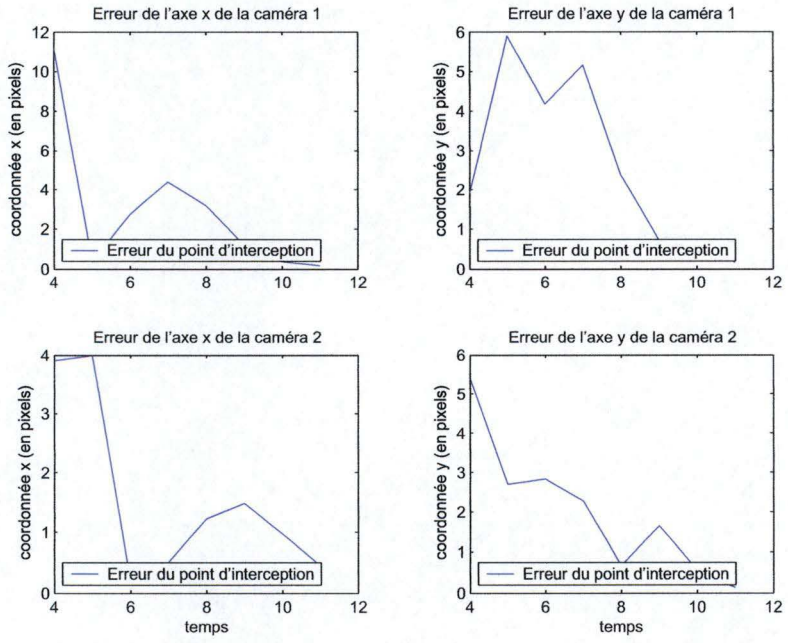


FIG. 10.6 – *Erreur de prédiction.*

10.8 Conclusion

Dans ce chapitre, nous avons vu une méthode simple qui est tout à fait exploitable dans la réalité. Le fait d'utiliser le temps comme élément déterminant de la prédiction, rend le modèle robuste au bruit. Cependant, ce modèle nécessite un certain nombre de points pour atteindre la précision souhaitée. Il faudra donc veiller à disposer d'un grand nombre d'échantillons par trajectoire.

Ce modèle applicable en pratique, fournit rapidement un point d'interception situé dans le voisinage du point réel. De plus, si nous comparons ce modèle à ceux rencontrés dans la littérature, nous voyons que nos trajectoires peuvent être quelconques. Ensuite, nous n'utilisons aucun matériel spécialisé. Remarquons encore que nous réalisons une prédiction dans l'image, ce qui constitue une approche tout à fait originale.

Conclusion

Ce mémoire retrace l'élaboration d'une série de techniques tentant de prédire le mieux possible une trajectoire balistique observée par un dispositif de vision stéréoscopique. Par ses objectifs, il constitue une approche originale du problème.

Pour rappel, le premier objectif était la réalisation de la prédiction dans les espaces images des caméras. Le second était l'utilisation de matériel non spécialisé, c'est-à-dire l'emploi de simples webcams non-synchronisées. Le troisième n'imposait qu'un minimum, voire aucune connaissance de la position des caméras, ni de la scène, c'est-à-dire qu'il n'exigeait aucun élément de référence, ni aucune information préalable sur le type de trajectoire. Le quatrième était l'identification de système.

Ce mémoire est donc en marge de ce qui existe actuellement dans la littérature. En abordant les points cités ci-dessus, il réalise un apport intéressant d'éléments nouveaux en supprimant un maximum de contraintes liées aux conditions d'expérimentations et en se servant d'un matériel non spécialisé.

Dans un premier temps, des méthodes dites neuro-mimétiques furent utilisées. D'autres méthodes, dont le fonctionnement est similaire, ont aussi été étudiées. En effet, comme nous l'avons vu, les algorithmes neuro-mimétiques employés sont des approximations d'algorithmes de réduction de l'erreur sur un ensemble de données statiques, comme la régression linéaire. Cela permet de remplacer, par exemple, la régression linéaire de notre dernier modèle par un algorithme α-LMS ou "Recursive Least Square".

Le dernier modèle construit dans le chapitre 10 constitue un apport particulier par rapport à la littérature et pour le laboratoire "TROP". En effet, il répond aux principales attentes du laboratoire et il élargit la gamme de problèmes rendue accessible dans la littérature. De plus, il offre de nouvelles perspectives de recherche.

Tout d'abord, ce modèle réalise la prédiction de la trajectoire balistique directement dans l'espace bidimensionnel de l'image, ce qui constitue l'objectif le plus important du laboratoire. De plus, prédire une série temporelle dans l'image est une problématique qui n'est pas abordée dans la littérature. Celle-ci présente des méthodes qui utilisent un espace cartésien reconstruit à partir des deux caméras.

Deuxièmement, notre modèle ne nécessite aucune connaissance de la disposition de la scène ni aucune calibration préalable du dispositif d'observation. Nous pouvons donc affirmer qu'il rencontre le souhait du laboratoire qui vise à développer des méthodes qui apprennent leur environnement de travail ou qui en sont indépendantes.

Cependant, notre modèle n'atteint pas un de nos objectifs initiaux : l'identification de système. Ce point demeure une perspective de recherche, mais c'est aussi le cas pour les modèles actuels de la littérature qui traitent de la prédiction de trajectoires balistiques. Nous avions fixé cet objectif comme objectif secondaire lors de l'élaboration de ce travail.

Par contre, remarquons que notre modèle peut prédire une trajectoire quelconque. Cela constitue un grand avantage par rapport aux autres travaux qui nécessitent des trajectoires presque parallèles aux plans des deux caméras. De plus, il ne réclame aucun matériel spécialisé, ce qui est encore différent des travaux, recensés dans la littérature, dans lesquels on exploite du matériel de haute performance. Son emploi peut, aussi, être aisément étendu pour des caméras dont le plan des axes focaux n'est plus perpendiculaire au vecteur de gravité.

En définitive, notre travail ouvre de nombreuses perspectives de recherches pour améliorer les idées présentées ou pour les compléter. Nous avons introduit l'utilisation de webcams et montré, dans le chapitre 8, leurs limitations. Cette technologie est intéressante pour le laboratoire dans la mesure où celui-ci essaie de remplacer le matériel spécialisé de vision par du matériel plus simple.

Nous avons aussi présenté un traitement d'images simple, qui pourra faire l'objet de beaucoup d'améliorations. Dans ce cadre, comme décrit au chapitre 5, il serait intéressant de voir si le nombre d'échantillons collectés par les webcams ne peut pas être étendu artificiellement. Cela permettrait, peut-être, de prédire plus rapidement la trajectoire. Par manque temps, cette idée d'étendre le nombre d'observations artificiellement n'a pas pu être explorée. Dès lors, cela reste une ouverture pour de nouvelles investigations.

Ce travail pourrait aussi servir de base pour réaliser l'interception d'une balle en temps réel par le bras robotique du laboratoire "TROP". La prédiction devra alors être secondée d'une technique d'interception, c'est-à-dire la décision et l'optimisation d'un point d'impact.

En conclusion, notre étude exploratoire constitue une approche originale par rapport à l'existant. Elle aborde des aspects non explorés par la littérature. Elle débroussaille le domaine des séries temporelles en montrant, notamment, les difficultés rencontrées en tentant de prédire une série temporelle particulière. Enfin, elle répond à la majeure partie des objectifs fixés initialement et elle ouvre de nouveaux horizons de recherches au laboratoire "TROP".

Annexe A

Modèle d'un objet uniformément accéléré

A.1 Introduction

Nous montrons, dans cette annexe, l'élaboration du modèle d'un objet uniformément accéléré, car celui-ci est utilisé dans le chapitre 3. Comme il est modélisé, nous ne tenons pas compte de force de frottement ni d'effet dû à une quelconque rotation, c'est-à-dire de Force de Magnus. Une représentation de ce modèle est illustré à la figure A.1.

A.2 Développement

Considérons un corps de masse m_1 à la surface de la terre. La terre attire cette masse avec une force de $9{,}81m_1$ Newtons. En effet, la force d'attraction gravifique terrestre, qui accélère la masse m_1 avec une accélération g, vaut :

$$F_{grav} = m_1 g = m_1 \frac{GM_T}{R^2} = m_1 \frac{6{,}67.10^{-11} 5{,}98.10^{24}}{(6{,}38.10^6)^2} \simeq m_1 9.81 \qquad \text{(A.1)}$$

où :

$G = 6{,}67.10^{-11}$ *newtons* $*$ m^2/kg^2 , *la constante de gravitation universelle,*
$M_T = 5{,}98.10^{24}$ *kg* , *la masse de la terre,*
$R = 6{,}38.10^6$ *m* , *le rayon de la terre.*

L'accélération est indépendante de la masse du corps considéré. Une bonne approximation de cette valeur, à la surface de la terre, vaut 9,81 m/s^2. La norme de cette accélération peut être considérée comme constante tant que l'objet ne s'élève pas trop haut. De même, la direction de cette accélération peut aussi être considérée comme constante tant que l'objet ne parcourt pas une trop grande distance à la surface de la terre. En première approximation, un objet lancé à la surface de la terre est donc soumis à une seule force dirigée vers le centre de la planète.

Démontrons l'équation de la trajectoire d'un objet uniformément accéléré :

Soient les scalaires $p(t)$ et $v(t)$, avec $\vec{l_i}$ vecteurs unitaires $i \in \{1,2\}$, tels que

$$\vec{p}(t) = p(t)\ \vec{l_1} \quad et \quad \vec{v}(t) = v(t)\ \vec{l_2}.$$

Par hypothèse, l'accélération est constante

$$\vec{a(t)} = \vec{a_0} = \vec{g} \qquad \forall t$$

et en $t = 0$, le point initial est $p(t = 0) = p_0$ et la vitesse initiale est $v(t = 0) = v_0$.

Sachant que

$$\vec{a} = \frac{d\vec{v}}{dt},$$

c'est-à-dire

$$a_x = \frac{dv_x}{dt} = \frac{d^2p_x}{dt^2}\ ,\ a_y = \frac{dv_y}{dt} = \frac{d^2p_y}{dt^2}\ ,\ a_z = \frac{dv_z}{dt} = \frac{d^2p_z}{dt^2},$$

nous obtenons :

$$dv_x(t) = a_0\ dt$$

qui devient après intégration

$$v_x(t) = a_0 t + constante.$$

Or, $v(t = 0) = v_0$, ce qui implique

$$v_0 = constante.$$

Donc,

$$v_x(t) = a_0 t + v_0 \qquad \forall t.$$

Avec le même raisonnement,

$$\vec{v} = \frac{d\vec{p}}{dt}$$

implique que

$$dp(t) = v_x(t)dt = (a_0 t + v_0)dt.$$

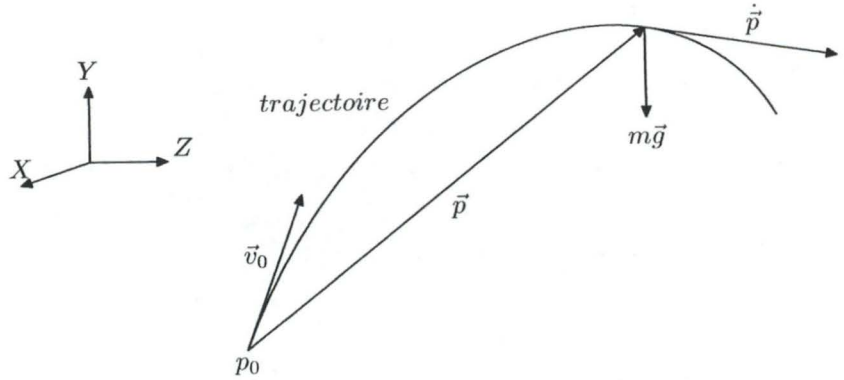


FIG. A.1 – *Modèle uniformément accéléré d'un projectile.*

Après intégration, nous avons

$$p(t) = \frac{a_0 t^2}{2} + v_0 t + constante$$

avec $p(t=0) = p_0$:

$$p(t) = \frac{a_0 t^2}{2} + v_0 t + p_0 \qquad \forall t.$$

En résumé, nous obtenons les équations suivantes:

$$\ddot{\vec{p}}(t) = \vec{g} \qquad \forall t, \qquad \text{(A.2)}$$

$$\dot{\vec{p}}(t) = \vec{v}_0 + \vec{g}t \qquad \forall t, \qquad \text{(A.3)}$$

$$\vec{p}(t) = \vec{p}_0 + \vec{v}_0 t + \frac{\vec{g}t^2}{2} \qquad \forall t \qquad \text{(A.4)}$$

avec la notation

$$\dot{\vec{p}} = \frac{d\vec{p}}{dt} = \vec{v}$$

et

$$\ddot{\vec{p}} = \frac{d\dot{\vec{p}}}{dt} = \frac{d^2\vec{p}}{dt^2} = \vec{a}$$

où :

$$\vec{p}_0 = la\ position\ de\ départ,$$
$$\vec{v}_0 = la\ vitesse\ de\ départ,$$
$$\vec{g} = l'accélération\ gravifique.$$

Il est à noter que nous utilisons une convention inhabituelle pour le repère orthonormé de l'espace cartésien. Comme schématisé sur la figure A.1, le vecteur de gravité

est inverse à l'axe Y et n'est pas dans le sens de Z comme habituellement utilisé. C'est en réalité un problème de convention avec le modèle d'une caméra, car pour une caméra, l'axe Z est utilisé comme axe optique et les axes X et Y pour le plan image. De plus, comme les axes X et Y sont dans le plan image, il est alors facile d'utiliser le passage d'un élément de la scène à sa représentation dans l'image dont les axes sont justement X et Y. Nous avons donc choisi d'utiliser le vecteur gravité $\vec{g} = (0, -g, 0)^T$.

Annexe B

Passage en temps discret

B.1 Introduction

Le passage en temps discret décrit dans cette annexe est de type stroboscopique, car il donne une relation entre les variables du système aux instants d'échantillonnage. Cette démonstration est tirée du livre "Computer-controlled systems" [1]. Le lecteur intéressé par davantage de détails se référera à cet ouvrage.

B.2 Développement

L'état au temps t, $t_k \leq t \leq t_{k+1}$, est obtenu en résolvant les équations [1] :

$$\dot{x} = Ax(t) + Bu(t), \tag{B.1}$$

$$y(t) = Cx(t) + Du(t). \tag{B.2}$$

Cela donne comme solution

$$x(t) = e^{A(t-tk)}x(t_k) + \int_{t_k}^{t} e^{A(t-s')}Bu(s')ds'.$$

En considérant que $u(s')$ est constant entre t_k et t, on a

$$x(t) = e^{A(t-tk)}x(t_k) + \int_{t_k}^{t} e^{A(t-s')}ds'Bu(t_k),$$

$$x(t) = e^{A(t-tk)}x(t_k) + \int_{0}^{t-t_k} e^{As}dsBu(t_k),$$

$$x(t) = \phi(t,t_k)x(t_k) + \tau(t,t_k)u(t_k).$$

Le vecteur d'état, au temps t, est une fonction linéaire de $x(t_k)$ et $u(t_k)$. Le système d'équations du système échantillonné devient :

$$x(t_{k+1}) = \phi(t_{k+1},t_k)x(t_k) + \tau(t_{k+1},t_k)u(t_k), \tag{B.3}$$

$$y(t_k) = Cx(t_k) + Du(t_k) \tag{B.4}$$

et nous avons

$$\phi(t_{k+1},t_k) = e^{A(t_{k+1}-t_k)} = e^{Ah},$$

$$\tau(t_{k+1},t_k) = \int_0^{t_{k+1}-t_k} e^{As} dsB = \int_0^h e^{As} dsB$$

avec $h = t_{k+1} - t_k$ le taux d'échantillonnage.

Annexe C

Développement du calcul de la profondeur

C.1 Introduction

Nous expliquons maintenant le calcul de la distance entre le centre optique C_1 d'une caméra et le plan perpendiculaire à l'axe optique de cette caméra, plan comprenant le point p. Le modèle sténopé, développé dans la section 4.3 du chapitre 4, est justement non-linéaire en raison de cette distance p_z, qui est la profondeur de l'objet centré en p.

Pour calculer cette distance, nous avons besoin des informations d'une seconde caméra observant le même point p. Nous considérons que les axes optiques des deux caméras sont situés dans un même plan. Nous allons donc calculer la distance p_z^1, en référence à la caméra n°1, comme schématisée à la figure C.2.

Sur cette figure, X_1 représente la distance entre le point p et l'axe optique de la première caméra. Ce segment est donc perpendiculaire à l'axe optique. Son image sur le plan image de la caméra, $\dot{X}_1$, représente la distance entre le centre du plan image (le plan π_1 à la figure C.1) et le point p projeté sur ce plan (le point m_1 à la figure C.1). Pour obtenir concrètement $\dot{X}_1$ avec une caméra, il faut soustraire la moitié de la résolution horizontale de l'image de la coordonnée horizontale du point observé, c'est-à-dire sur l'axe U, puis il ne faut pas oublier de convertir cette distance exprimée en pixels en une distance en mètre par exemple. Pour la deuxième caméra, X_2 et $\dot{X}_2$ sont analogues aux X_1 et $\dot{X}_1$ de la première caméra, sauf que leurs sens sont opposés. Il faut soustraire la coordonnée horizontale du point observé de la moitié de la résolution horizontale de l'image pour obtenir $\dot{X}_2$.

Au regard de la figure C.2, nous observons que p_z^1 est la projection de D_1 sur l'axe optique de la caméra n°1. Nous pouvons donc connaître p_z^1 par un produit scalaire de deux vecteurs ($\vec{p}$ et $\vec{f}$). Il nous faut cependant avoir connaissance de la distance du segment D_1, c'est-à-dire des coordonnées du point p si nous considérons un repère bidimensionnel orthonormé centré en C_1. Pour déterminer les coordonnées du point p, nous allons calculer l'intersection des droites D_1 et D_2. La droite D_1 passe par deux points connus : C_1 et l'intersection de la droite $\dot{X}_1$ et de D_1. Nous pouvons donc établir son équation. Nous appliquerons le même raisonnement avec la droite D_2, sachant que C_2 est à une distance l du centre du repère.

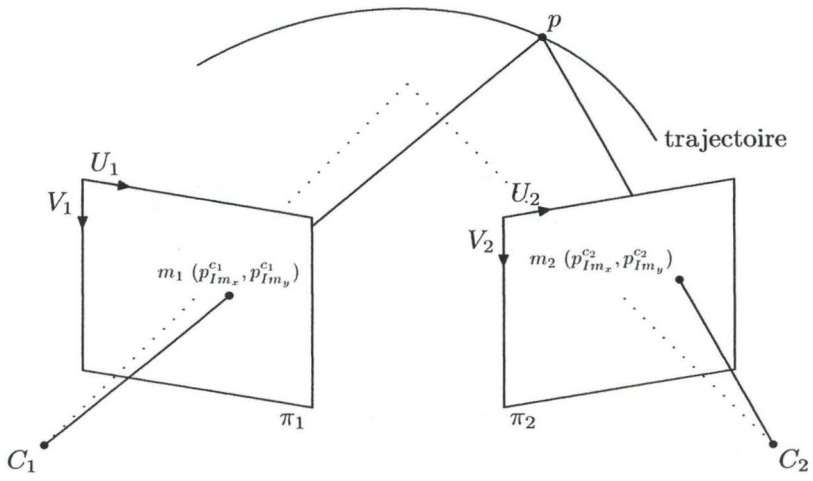


FIG. C.1 – *Schéma représentant la scène où l'objet p est dans le champ de vision des deux caméras. C_1 et C_2 sont les centres optiques des deux caméras. π_1 et π_2 représentent les plans images. Les droites en pointillé représentent les axes optiques des caméras.*

C.2 Développement

Prenons comme centre d'un repère, le centre optique C_1 de la caméra numéro 1. Soit un point $p = (x_p,y_p)$ dans ce repère. Construisons un vecteur unitaire $\vec{f}$ dans la direction et le sens de $\vec{f}$ de la caméra numéro 1 . Nous avons

$$p_z^1 =< \vec{p}.\vec{f} >= x_p cos(\alpha_1) + y_p sin(\alpha_1). \tag{C.1}$$

Cherchons maintenant l'expression de x_p et de y_p. Il nous suffit pour cela de calculer l'intersection des droites D_1 et D_2 :

$$D_1 \equiv \forall x \; \exists y \; tq \quad ay = bx, \tag{C.2}$$

$$D_2 \equiv \forall x \; \exists y \; tq \quad cy = d\,(x - l)\,. \tag{C.3}$$

En identifiant a, b, c et d :

$$D_1 \equiv \forall x \; \exists y \; tq \quad \underbrace{\Big(f cos(\alpha_1) + \dot{X}_1 sin(\alpha_1)\Big)}_{a} y = \underbrace{\Big(f sin(\alpha_1) - \dot{X}_1 cos(\alpha_1)\Big)}_{b} x$$

et

$$D_2 \equiv \forall x \; \exists y \; tq \quad \underbrace{\Big(-f cos(\alpha_2) - \dot{X}_2 sin(\alpha_2)\Big)}_{c} y = \underbrace{\Big(f sin(\alpha_2) - \dot{X}_2 cos(\alpha_2)\Big)}_{d} (x - l)\,.$$

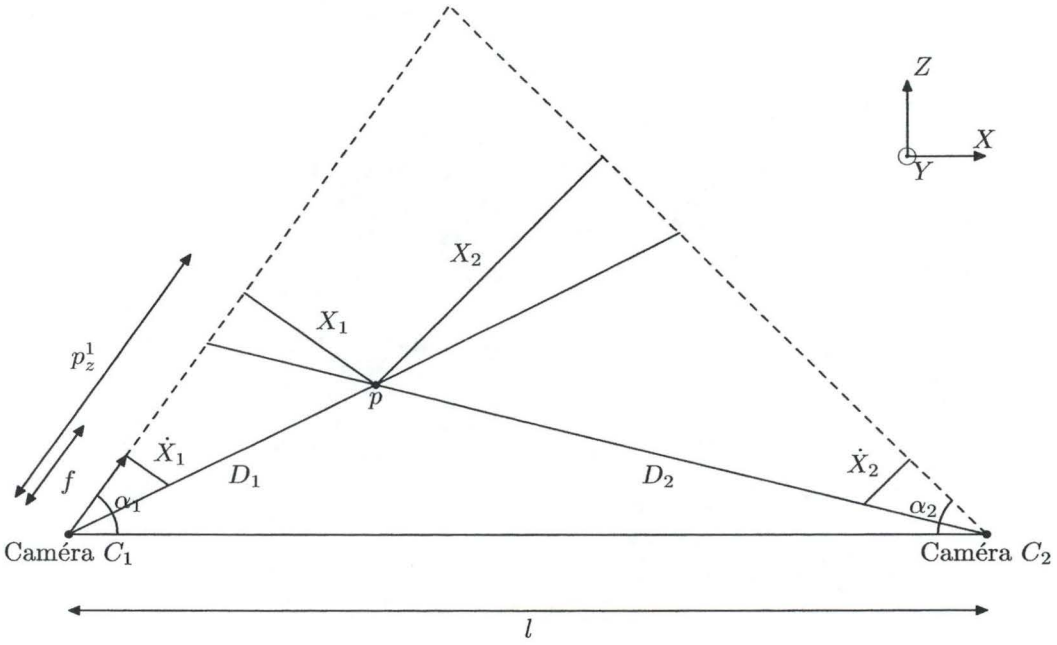


FIG. C.2 – *Représentation de la scène, dans le plan des axes optiques, permettant le calcul de la profondeur p_z^1 du point p par rapport à la caméra n°1 dont le centre optique est en C_1. Les segments en pointillé représentent les axes optiques des caméras.*

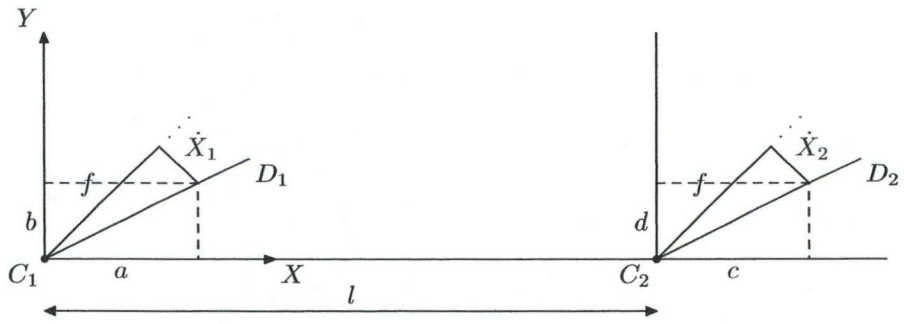


FIG. C.3 – *Illustration du raisonnement.*

Nous avons donc, comme intersection, des deux droites,

$$x_p = \frac{-dla}{(cb - da)} \; et \; y_p = \frac{-dlb}{(cb - da)} \tag{C.4}$$

et en injectant cela dans l'équation C.1,

$$p_z^1 = -l\frac{dacos(\alpha_1) + dbsin(\alpha_1)}{(cb - da)} \tag{C.5}$$

avec :

$$
\begin{aligned}
cb - da = & \\
& \Big(-f cos(\alpha_2) - \dot{X}_2 sin(\alpha_2)\Big)\Big(f sin(\alpha_1) - \dot{X}_1 cos(\alpha_1)\Big) \\
& - \Big(f sin(\alpha_2) - \dot{X}_2 cos(\alpha_2)\Big)\Big(f cos(\alpha_1) + \dot{X}_1 sin(\alpha_1)\Big) \qquad \text{(C.6)} \\
= \; & -f\dot{X}_2 sin(\alpha_2) sin(\alpha_1) + \dot{X}_1\dot{X}_2 sin(\alpha_2) cos(\alpha_1) \\
& -f^2 cos(\alpha_2) sin(\alpha_1) + f\dot{X}_1 cos(\alpha_2) cos(\alpha_1) \\
& -f^2 sin(\alpha_2) cos(\alpha_1) - f\dot{X}_1 sin(\alpha_2) sin(\alpha_1) \\
& +f\dot{X}_2 cos(\alpha_1) cos(\alpha_2) + \dot{X}_1\dot{X}_2 cos(\alpha_2) sin(\alpha_1) \qquad \text{(C.7)} \\
= \; & f\dot{X}_2 cos(\alpha_1 + \alpha_2) + \dot{X}_1\dot{X}_2 sin(\alpha_1 + \alpha_2) \\
& -f^2 sin(\alpha_1 + \alpha_2) + f\dot{X}_1 cos(\alpha_1 + \alpha_2), \qquad \text{(C.8)}
\end{aligned}
$$

$$
\begin{aligned}
da &= \Big(f sin(\alpha_2) - \dot{X}_2 cos(\alpha_2)\Big)\Big(f cos(\alpha_1) + \dot{X}_1 sin(\alpha_1)\Big) \qquad \text{(C.9)} \\
&= f^2 sin(\alpha_2) cos(\alpha_1) + f\dot{X}_1 sin(\alpha_2) sin(\alpha_1) \\
&\quad - f\dot{X}_2 cos(\alpha_2) cos(\alpha_1) - \dot{X}_1\dot{X}_2 cos(\alpha_2) sin(\alpha_1), \qquad \text{(C.10)}
\end{aligned}
$$

$$
\begin{aligned}
db &= \Big(f sin(\alpha_2) - \dot{X}_2 cos(\alpha_2)\Big)\Big(f sin(\alpha_1) - \dot{X}_1 cos(\alpha_1)\Big) \qquad \text{(C.11)} \\
&= -f^2 sin(\alpha_2) sin(\alpha_1) + f\dot{X}_2 cos(\alpha_2) sin(\alpha_1) \\
&\quad f\dot{X}_1 sin(\alpha_1) cos(\alpha_1) - \dot{X}_1\dot{X}_2 cos(\alpha_2) cos(\alpha_1). \qquad \text{(C.12)}
\end{aligned}
$$

Développons maintenant le numérateur de l'équation C.5 :

$$
\begin{aligned}
& da cos(\alpha_1) + db sin(\alpha_1) = \\
& sin(\alpha_2) cos^2(\alpha_1) f^2 + sin(\alpha_2) sin(\alpha_1) cos(\alpha_1) f\dot{X}_1 \\
& - cos(\alpha_2) cos^2(\alpha_1) f\dot{X}_2 - cos(\alpha_2) cos(\alpha_1) sin(\alpha_1)\dot{X}_1\dot{X}_2 \\
& + sin(\alpha_2) sin^2(\alpha_1) f^2 - cos(\alpha_2) sin^2(\alpha_1) f\dot{X}_2 \\
& + cos(\alpha_2) cos(\alpha_1) sin(\alpha_1)\dot{X}_1\dot{X}_2 - sin(\alpha_2) sin(\alpha_1) cos(\alpha_1) f\dot{X}_1
\end{aligned}
$$

et simplifions cette dernière équation :

$$
da cos(\alpha_1) + db sin(\alpha_1) = f^2 sin(\alpha_2) - f\dot{X}_2 cos(\alpha_2). \qquad \text{(C.13)}
$$

De C.5, C.8 et C.13, nous obtenons :

$$p_z^1 = \frac{l(f\dot{X}_2 cos(\alpha_2) - f^2 sin(\alpha_2)}{(\dot{X}_1 + \dot{X}_2) f cos(\alpha_1 + \alpha_2) + (\dot{X}_1 \dot{X}_2 - f^2) sin(\alpha_1 + \alpha_2)}. \tag{C.14}$$

Avec la notation $p_{Im_x}^{c1}$ pour la caméra numéro 1 et $p_{Im_x}^{c2}$ pour la caméra numéro 2 (Cfr. Eq. 4.2, 4.3 du chapitre 4):

$$p_z^1 = \frac{l(f(-\frac{p_{Im_x}^{c2}}{\Psi_x} + \frac{u_{max}}{2\Psi_x}) cos(\alpha_2) - f^2 sin(\alpha_2))}{(\frac{p_{Im_x}^{c1}}{\Psi_x} - \frac{p_{Im_x}^{c2}}{\Psi_x}) f cos(\alpha) + ((\frac{p_{Im_x}^{c1}}{\Psi_x} - \frac{u_{max}}{2\Psi_x})(-\frac{p_{Im_x}^{c2}}{\Psi_x} + \frac{u_{max}}{2\Psi_x}) - f^2) sin(\alpha)} \tag{C.15}$$

avec :

f = la distance focale identique sur les deux caméras,
l = la distance entre les deux centres optiques C_1 et C_2,
α_1 = l'angle entre la base et l'axe optique de la caméra n°1,
α_2 = l'angle entre la base et l'axe optique de la caméra n°2,
α = $\alpha_1 + \alpha_2$,
$p_{Im_x}^{c1}$ = $\Psi_x \dot{X}_1 + \frac{u_{max}}{2}$,
$p_{Im_x}^{c2}$ = $-\Psi_x \dot{X}_2 + \frac{u_{max}}{2}$.

Dans le cas où nous plaçons les axes optiques des caméras parallèlement, $\alpha_1 = \frac{\pi}{2}$ et $\alpha_2 = \frac{\pi}{2}$, nous obtenons la simplification suivante :

$$p_z^1 = \frac{lf}{\dot{X}_1 + \dot{X}_2} = \frac{lf}{\frac{p_{im_x}^{c1}}{\Psi_x} - \frac{p_{im_x}^{c2}}{\Psi_x}}. \tag{C.16}$$

Comme déjà énoncé plus haut, les équations C.15 et C.16 ne sont valables que dans le cas où les axes optiques des deux caméras ne sont pas gauches, c'est-à-dire qu'ils peuvent être compris dans un même plan. Cette hypothèse est peu contraignante et largement utilisée en pratique.

Nous pouvons donc, en connaissant la position d'un objet dans les deux images, les angles α_1 et α_2 (des caméras) et la distance l, déterminer le dénominateur p_z^1 sous-jacent au modèle des caméras (Cfr. Eq. 4.2 et 4.3 du chapitre 4).

Bibliographie

[1] Karl J. Astrom and Bjorn Wittenmark. *Computer-controlled systems. Theory and design.* Prentice Hall, Upper Saddle River, New Jersey 07458, 1997. ISBN 0-13-314899-8.

[2] Rembrandt Bakker, Jaap C. Schouten, Cor M. van den Bleek, and C. Lee Giles. Neural learning of chaotic dynamics: The error propagation algorithm. In *IEEE World Conf. on Comp. Intelligence*, page 2483, 1998.

[3] S. Bengio, F. Fessant, and D. Collobert. Use of modular architectures for time-series prediction. *Neural Processing Letters*, 3(2):101–106, 1996.

[4] Edmond Boyer and Peter Sturm. *Traité IGAT : Synthèse d'images géographiques*, chapter Modélisation à partir d'images, pages 57–89. Hermes, 2002.

[5] Martin Brown and Chris Harris. *Neurofuzzy Adaptive Modelling and Control.* Prentice Hall, 1994. ISBN 0-13-134453-6.

[6] J.L. Buessler. Architectures neuro-mimétiques modulaires. application à l'asservissement visuel de systèmes robotiques., 1999.

[7] E. S. Chng, S. Chen, and B. Mulgrew. Gradient radial basis function networks for nonlinear and nonstationary time series prediction. *IEEE Transactions on Neural Networks*, 7(1):190–194, January 1996.

[8] Tomasz J. Cholewo and Jacek M. Zurada. Sequential network construction for time series prediction. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 2034 – 2039, 1997.

[9] N. Davey, S.P. Hunt, and R.J. Frank. Time series prediction and neural networks.

[10] G. Dreyfus, J.-M Martinez, M. Samuelides, M. B. Gordon, F. Badran, S. thiria, and L. Hérault. *Réseaux de neurones. Méthodologie et applications.* Algorithmes. Editions Eyrolles, 61, BLD Saint-Germain 75240 Paris cedex 05, Janvier 2002. ISBN 2-212-11019-7.

[11] Rimon Elias and Robert Laganière. Projective geometry for three-dimensional computer vision. Technical report, School of Information Technology and Engineering University of Ottawa, Ottawa.

[12] Christian Feldbauer, Franz Pernkopf, and Erhard Rank. Adaptive filters, a tutorial for the course computational intelligence. Signal Processing and Speech Communication Laboratory.

[13] Simon Haykin. *Neural networks. A comprehensive foundation.* Prentice Hall, Upper Saddle River, New Jersey 07458, 1999. ISBN 0-13-273350-1.

[14] Nicholas John Hollinghurst. *Uncalibrated Stereo and Hand-Eye Coordination.* PhD thesis, University of Cambridge, January 1997.

[15] W. Hong and J. Slotine. Experiments in hand-eye coordination using active vision, 1995.

[16] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski. Time series prediction using RSOM with local linear modesl. Technical Report B15, Espoo, Finland, 1997.

[17] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski. Recurrent SOM with local linear models in time series prediction. In *6th European Symposium on Artificial Neural Networks. ESANN'98. Proceedings. D-Facto, Brussels, Belgium*, pages 167–72, 1998.

[18] A. Lendasse, E. de Bodt, and M. Verleysen. Estimation de la dimension intrinsèque d'une série temporelle et prédiction par une méthode de projection, 2002.

[19] A. Lendasse, E. de Bodt, V. Wertz, and M. Verleysen. Non-linear financial time series forecasting - application to the bel 20 stock market index, 2000.

[20] A. Lendasse, J. Lee, E. De Bodt, V. Wertz, and M. Verleysen. Dimension reduction of technical indicators for the prediction of financial time series - application to the bel 20 market index. *European Journal of Economic and Social Systems*, 15(2), 2001.

[21] A. Lendasse, J. Lee, E. de Bodt, V. Wertz, and M. Verleysen. Input data reduction for the prediction of financial time series, 2001.

[22] A. Lendasse, J. Lee, V. Wertz, and M. Verleysen. Time series forecasting using cca and kohonen maps - application to electricity consumption, 2000.

[23] Amaury Lendasse, Vincent Wertz, and Michel Verleysen. Model selection with cross-validations and bootstraps - application to time series prediction with rbfn models, 2003.

[24] Uros Lotric and Andrej Dobnikar. Wavelet based smoothing in time series prediction with neural networks.

[25] Simona Malaroiu, Kimmo Kiviluoto, and Erkki Oja. Time series prediction with independent component analysis, 1999.

[26] Christopher M.Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995. ISBN 0-19-853864-2.

[27] Roger Mohr and Bill Triggs. *Projective Geometry for Image Analysis A Tutorial given at ISPRS.* Vienna, July 1996.

[28] Akio Namiki and Masatoshi Ishikawa. Robotic catching using a direct mapping from visual information to motor command, 2003.

[29] N. Otsu. A threshold selection method from gray-level histograms. In *Transactions on Systems, Man, and Cybernetics*, volume 9, No. 1, pages 62–66. IEEE, 1979.

[30] Hong Pi and Carsten Peterson. Finding the embedding dimension and variable dependencies in time series. *Neural Computation*, 6(3):509–520, May 1994.

[31] J. C. Principe, L. D. Wang, and M. A. Motter. Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control. In *Proc. IEEE*, pages 2240–2258, 1998.

[32] M. Riley, A. Ude, and C. Atkeson. Methods for motion generation and interaction with a humanoid robot: Case studies of dancing and catching, 2000.

[33] Marcia Riley and Christopher Atkeson. Robot catching.

[34] Helge Ritter, Thomas Martinetz, and Klaus Schulten. *Neural Computation and Self-Organizing Maps.* Addison-Wesley, New York, 1992.

[35] Helge Ritter, Thomas Martinetz, and Klaus Schutlen. *Neural Computation and Self-Organisation - An Introduction.* Addison-Wesley Publishing Company, 1992. ISBN 0-201-55442-9.

[36] David Roussel. *Reconstruction de courbes et de surfaces 3d en stéréo-acquisition.* PhD thesis, Université Paris XI, Paris, 1999.

[37] Zaiyong Tang and Paul Fishwick. Feed-forward neural nets as models for time series forecasting. Technical Report 91-008.

[38] Geetha K. Thampi, Jose C. Principe, Mark A. Motter, JeongHo Cho, and Deniz Erdogmus. Adaptive inverse control using som based multiple models, 2002.

[39] Geetha K. Thampi, Jose C. Principe, Mark A. Motter, JeongHo Cho, and Jing Lan. Identification of aircraft dynamics using a som and local linear models, 2002.

[40] Geetha K. Thampi, Jose C. Principe, Mark A. Motter, JeongHo Cho, and Jing Lan. Multiple model based flight control design, 2002.

[41] S. Haidacher G. Schreiber I. Schaefer M. Hähnle G. Hirzinger U. Frese, B. Bäuml. Off-the-shelf vision for a robotic ball catcher. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui*, pages 1623 – 1629, 2001.

[42] M. Verleysen. The explanatory power of artificial neural networks. *The explanatory power of models*, 2002.

[43] Juha Vesanto. Using the SOM and local models in time-series prediction. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4–6*, pages 209–214. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.

[44] Hiroshi Wakuya and Jacek M. Zurada. Bi-directional computing architecture for time series prediction. *Neural Networks*, 14(9):1307–1321, 2000.

[45] Bernard Widrow and Michael Lehr. Thirty years of adaptive neural networks: Perceptron, madaline, and backpropagation. In *Proceedings of the IEEE*, volume 78, No. 9, page 1415. IEEE, 1990.

[46] Bernard Widrow and Eugene Walach. *Adaptive inverse control.* Prentice Hall, Upper Saddle River, New Jersey 07458, 1996. ISBN 0-13-005968-4.

[47] Patrice WIRA. Réseaux neuromimétiques, modularité et statistiques - estimation du mouvement pour l'asservissement visuel de robots, 2002.

[48] D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1329, 1998.

Index

Adaline, 1, 18, 24–33
Alpha-LMS, 31
Analyse
  des sources de problèmes, 109
  de sensibilité
    au bruit, 106
    au frottement, 60
Annexes, 137–147
Apport, 135
Apprentissage, 24
  supervisé, 25
Asservissement visuel, 1, 6

Balle, 53, 79
Bibliographie, 148–152
Bras robotique, 5
Bruit
  d'intégration, 110–111
  de pixélisation, 109
  de traitement d'images, 109–110

Calcul de la profondeur, 143–147
Calibration
  automatique, 100
Caméras, 5, 70
  perspectives, 64
Carte auto-organisatrice, voir SOM
Conclusion, 135–136
Contraintes, 15

Dendrites, 20
Descente de gradient, 27
Dilatation morphologique, 77
Discrétisation, 40
Disparité, 83
Durée d'intégration, 79
Désynchronisation, 111–114
Détection d'objet, 73

Effecteur, 5
Ensemble d'apprentissage, 24, 87
Erosion morphologique, 77
Espace de travail, 1, 5

Fenêtre
  d'entrée, 8
  temporelle, 8
Filtre de Kalman, 12
Fonction
  d'activation
    linéaire, 23
    sigmoïde, 23
    à seuil, 23
  d'apprentissage, 87
  de performance, 25
Force de frottement, 53–54

Groupe TROP, 1

Identification de système dynamique, 1
Image
  Conversion, voir Seuillage
  Soustraction, 76
  Traitement, 6, 73–82
Ingénierie, 1

Laboratoire MIPS, 1
Lampe stroboscopique, voir stroboscope
Least Mean Square (LMS), 18, 28–32
Limitations, 2
Littérature, 15

Matrice d'autocorrélation, 26
Matériel
  non spécialisé, 15
  spécialisé, 14
Mean Squared Error (MSE), 25
Modèle

balistique, 40
caméras non-parallèles, 97
balistique avec frottement, 53
d'un objet uniformément accéléré, 137
de prédiction
balistique avec frottement, 53
balistique simple, 39
multi-polynômial de prédiction, 125
neuro-mimétique
balistique, 42
balistique avec frottement, 54
d'apprentissage balistique, 85
pinhole, 64–66
polynômial de prédiction dans l'image, 117
sténopé, 64–66

Neurone
biologique, 20
formel, 22
Normalised Least Mean Square, 31

Ouverture morphologique, 77

Passage en temps discret, 41, 141–142
Perceptron, 18
Plate-forme robotique, 1, 5–7
Polynôme
prédiction, 104
Post-synchronisation, 78–79
Problème
de mise en oeuvre, 9
de validation, 9
Problématique, 5, 7
Projectile, 62
Projection perspective, 64

Recursive Least Square (RLS), 12, 32–33
Région d'intérêt, 6
Régression
linéaire, 27
linéaire multiple, 130–131
polynômiale, 12, 118–120
Réseaux de neurones, 34–37
caractéristiques, 35
multi-couches acycliques, 34
récurrents, 35
simple-couche acycliques, 34

Seuillage, 76
SOM, 7, 8, 19
Stimulus, 20
Stroboscope, 79
Surface de performance, 25–28
Synapses, 21
Synchronisation, 70, 79
Système
de vision, 5
distribué, 5
dynamique, 10
linéaire, 40
nerveux, 20
Séries temporelles, 1, 8–11

Taille des pixels, 85
Taux
d'apprentissage, 28, 31
d'échantillonnage, 6, 70
Techniques
de régressions, 8
à modèle unique, 8
à modèles locaux, 8
à modèles multiples, 9
Traitement d'images, 2, 73–82
Trajectoires balistiques, 1, 40

Vecteur
d'intercorrélation, 26

des poids, 24
Vision
perspective, 2, 64
stéréoscopique, 6

Webcams, 70